embesso®
embedded software solutions

# H8-Tiny

# TinyModule (TiMo)
# TinyBasicBoard (TiBBo)

# for Hitachi H8/36XX

User Manual

# PREFACE

Product Warranty
The warranty period against defects in materials and workmanship are as set out in the accompanying Customer Information sheet.

Limitation of Warranty
The foregoing warranty does not cover damage caused by fair wear and tear, abnormal storage conditions, incorrect use, accidental misuse, neglect, corruption, misapplication, addition or modification or by the use with other hardware or software, as the case may be, with witch the product is incompatible. No warranty of fitness for a particular purpose is offered. The user assumes the entire risk of using the product. Any liability of embesso GmbH is limited exclusively to the replacement of defective material or workmanship.

Trademarks
All brand or product names used in this manual are trademarks or registered trademarks of their respective companies or organizations.
Microsoft is a registered trademark and Windows and Windows NT are trademarks of Microsoft Corporation. IBM is a registered trademark of International Business Machines Corporation.

Cautions
This document may be, wholly or partially, subject to change without notice. All rights reserved. No one is permitted to reproduce or duplicate, in any form, a part or this entire document without embesso GmbH written permission.

Restrictions
Please refer to the restrictions of all components and tool suppliers.

Hardware Considerations
Grounding: This hardware is designed for use with equipment that is fully grounded. Ensure that all equipment used is appropriately grounded. Failure to do so could lead to danger for the operator or damage the equipment.
Electrostatic Discharge Precautions: This hardware contains devices that are sensitive to electrostatic discharge. Ensure appropriate precautions are observed during handling and accessing connections. Failure to do so could result in damage to the equipment.
Electromagnetic Compatibility: This product can cause radio frequency noise when used in the residential area. In such cases the user of the equipment my be required to take appropriate countermeasures under his responsibility.

Support
Support by EMBESSO GmbH is provided only for the supplied hardware. Any software tools are supported from their supplier. Please notice that maybe some software tools coming with this kit are only unsupported freeware and no support will be given.
For embesso support please contact: support@embesso.com

# Table of Contents

PREFACE ..... 2
Table of Contents ..... 3
1 TinyModule (TiMo) ..... 5
1.1 Hardware description ..... 5
1.2 Board overview ..... 6
1.3 Crystal socket ..... 7
1.4 Jumpers ..... 7
1.5 Connectors ..... 7
1.6 Start-Up instructions ..... 9
1.7 Schematics ..... 10
2 TinyBasicBoard (TiBBo) ..... 11
2.1 Hardware Overview ..... 11
2.2 Features ..... 12
2.3 Board overview ..... 13
2.4 Jumpers and switches ..... 14
2.5 Connectors ..... 17
2.6 Start-Up instructions ..... 19
2.7 Schematics ..... 20
2.8 Board layout ..... 21
3 Development Environment ..... 22
3.1 Software Tools ..... 22
3.2 Create a program using IAR-EWH8 ..... 24
3.3 Create a program using HEW ..... 32
3.4 Download the code using FDT ..... 38
3.5 Workflow ..... 46
4 Examples ..... 47
4.1 Timer_A ..... 48
4.2 SCI ..... 49
4.3 PWM+SCI ..... 51
Appendix A: CD-R content ..... 53

# 1 TinyModule (TiMo)

## *1.1 Hardware description*

Tiny Modules contain a microcontroller from Hitachi's H8/Tiny series. Different derivates are available (see table). The modules can be used to find out which of them is the best choose for your project. Or you can use different types while prototyping then for the target product. Together with the TinyBasicBoard (TiBBo) Hard- and Software developing is easy to do. When your target periphery board is available simply use a Tiny Module as the processor module for checking out your target hardware before doing your own final board design.

The modules contain the following parts:

H8/Tiny 36XX microcontroller (see table)
main quartz with socket
sub clock quartz (32.768 kHz)
64 pins with all controller signals
E10T-Debugger connector

Carefully remove the board from the shipping carton. Check first if there are any damages before power on the module.

Available modules

| Code | Flash/RAM | Options |
|---|---|---|
| Timo3664F | 32k / 2k | |
| Timo3664N | 32k / 2k | 512 Byte internal EEPROM |
| Timo3672F | 16k / 2k | without subclock |
| Timo3687G | 56k / 4k | POR / RTC / 2nd USART |
| Timo3694G | 16k / 2k | POR |

## *1.2 Board overview*

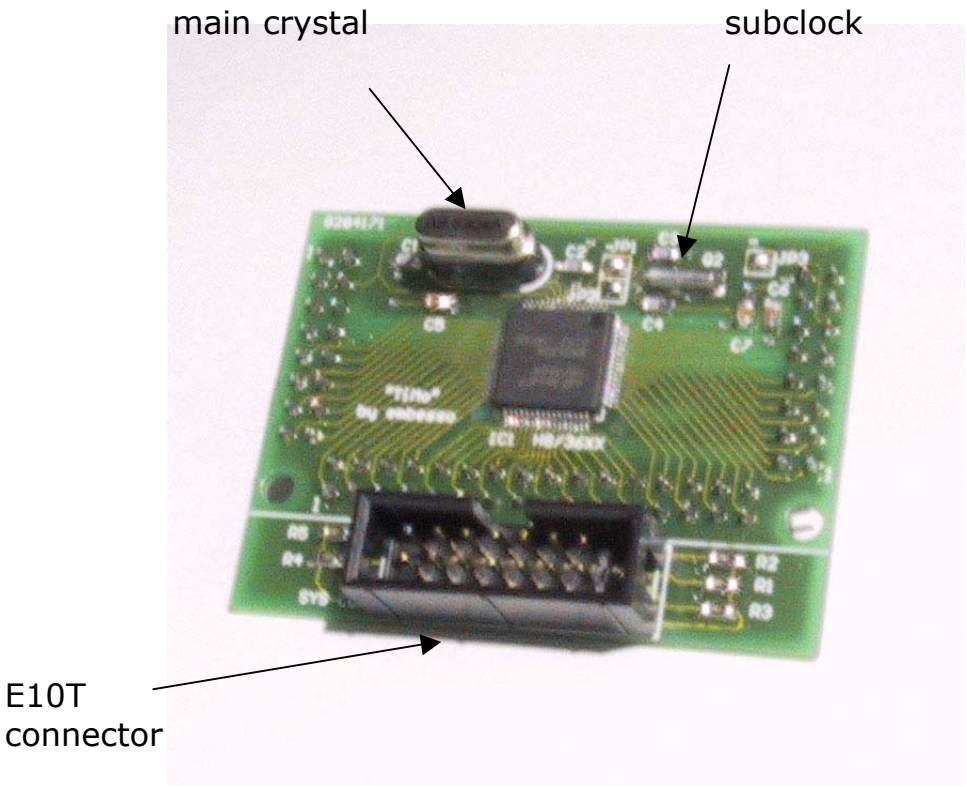



Figure 1 Board overview

## 1.3 Crystal socket

Main crystal (Q1) can be changed to another frequency. Therefore a crystal socket is provided. Please refer to the microcontroller hardware documentation for recommended devices. The crystal type should be a HC49 / HC49U type. Eventually change the capacitors C6/C7 if necessary. If you change the crystal frequency take care on possible changes by flash download tool (see FDT manual).

## 1.4 Jumpers

There are three soldering jumpers. JP1/JP2 are for using subclock. JP3 is to connect AVcc with Vcc.

| JP1 / JP2 | Operation |
|---|---|
| Closed (default) | Subclock enable |
| Open | Subclock disable |

| JP3 | Operation |
|---|---|
| Closed (default) | AVcc = Vcc |
| Open | AVcc = open |

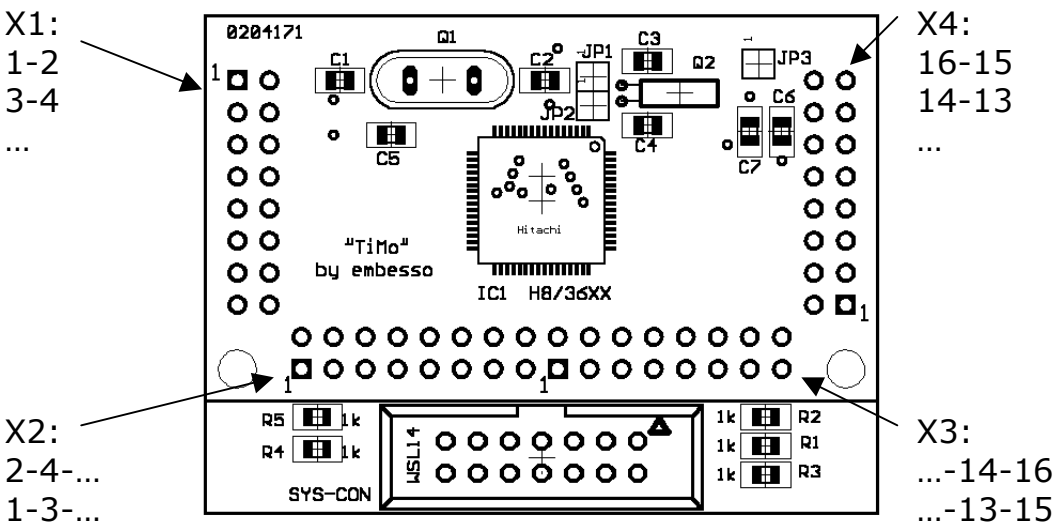
## 1.5 Connectors

X5, E10T debug connector (Sys-Con)

| X1, Pin | µC-pin | Operation |
|---|---|---|
| 1 | P87 | Debug pin |
| 5 | P86 | Debug pin |
| 7 | /NMI | /NMI |
| 11 | P85 | Debug pin |
| 13 | /RESET | reset control |
| 8 | Vcc | |
| 2,4,6,10,12,14 | GND | |
| 3,9 | n.c. | Not connected |

X1, X2, X3, X4 connectors with microcontroller signals

| X1 | | X2 | | X3 | | X4 | |
|---|---|---|---|---|---|---|---|
| PIN | Operation | PIN | Operation | PIN | Operation | PIN | Operation |
| 1 | GND | 1 | P12 | 1 | P85 | 1 | P31 |
| 2 | n.c. | 2 | P56 | 2 | P86 | 2 | P30 |
| 3 | n.c. | 3 | P57 | 3 | P87 | 3 | PB3 |
| 4 | VCC | 4 | P74 | 3 | P20 | 4 | PB2 |
| 5 | P50 | 5 | P75 | 5 | P21 | 5 | PB1 |
| 6 | P51 | 6 | P76 | 6 | P22 | 6 | PB0 |
| 7 | P34 | 7 | P24 | 7 | P23 | 7 | PB4 |
| 8 | P35 | 8 | P63 | 8 | P70 | 8 | PB5 |
| 9 | P36 | 9 | P62 | 9 | P71 | 9 | PB6 |
| 10 | P37 | 10 | P61 | 10 | P72 | 10 | PB7 |
| 11 | P52 | 11 | /NMI | 11 | P14 | 11 | AVCC |
| 12 | P53 | 12 | P60 | 12 | P15 | 12 | n.c. |
| 13 | P54 | 13 | P64 | 13 | P16 | 13 | n.c. |
| 14 | P55 | 14 | P65 | 14 | P17 | 14 | VCL |
| 15 | P10 | 15 | P66 | 15 | P33 | 15 | /RES |
| 16 | P11 | 16 | P67 | 16 | P32 | 16 | GND |



## *1.6 Start-Up instructions*

### 1.6.1 Installing the TiMo

Installing the TiMo requires a TinyBasicBoard or a user target board. Power down target board and put in the Tiny module. Check if all pins are connected! Then power up the target board.

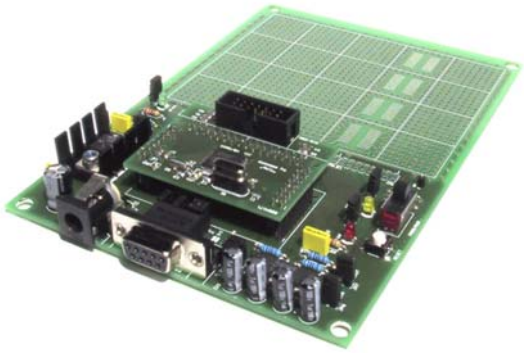
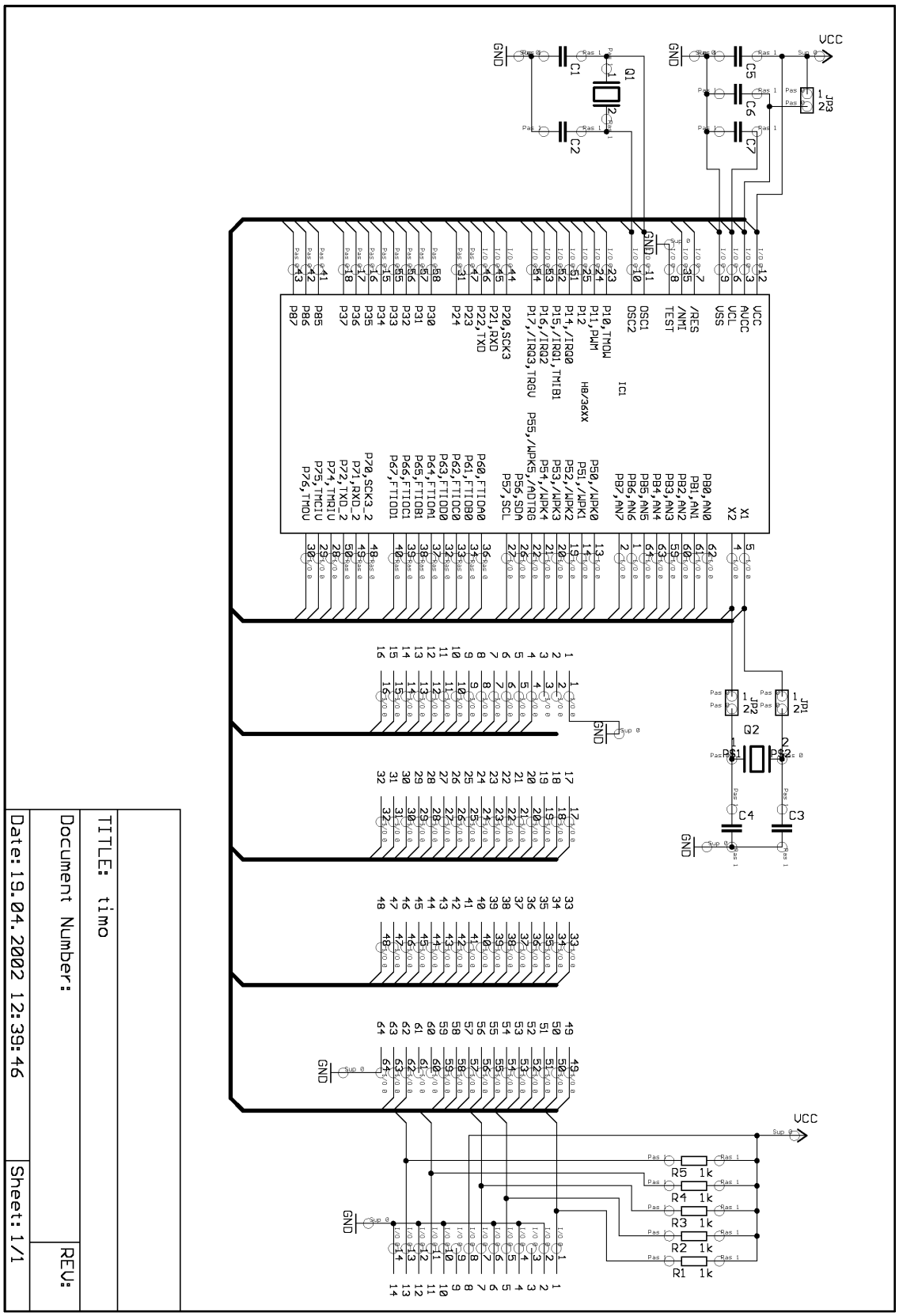


Figure 2 shows how to connect the TiMo to a TiBBo

## *1.7 Schematics*

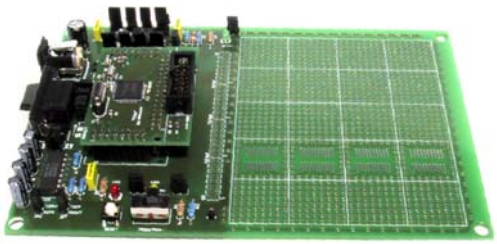


# 2 TinyBasicBoard (TiBBo)

## *2.1 Hardware Overview*

TiBBo is a basic board for Hard- and Software development with Tiny modules containing Hitachi's H8/Tiny microcontroller series. It can take one embesso TinyModule (TiMo) and provides development functionality like software download to flash, power supply, reset circuit and more. In addition it contains a great wire wrap field for user applicable periphery while development phase.

The TiBBo contains the following parts:

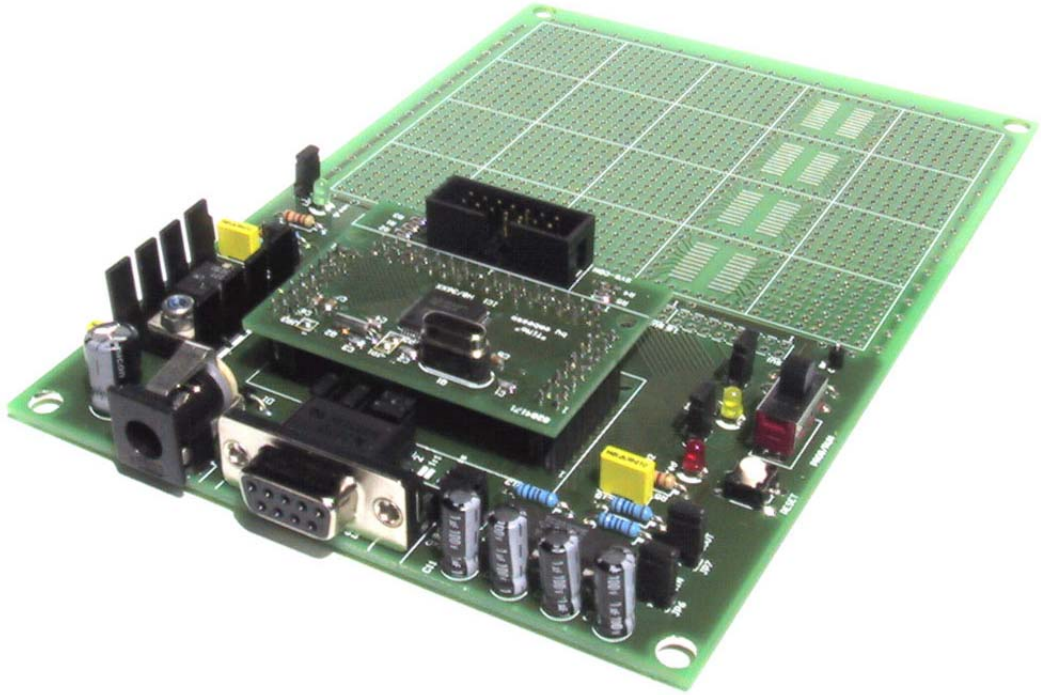


TinyBasicBoard (TiBBO)
+5V/3V3 voltage regulator
SubD9 connector and RS232 level converter
Reset button
Run/Program switch
4 plastic feet

Additionally there is a kit with a RS232 cable (1.8m, DSub9, male-female), an AC/DC Power Supply and a CD-R with some development tools and software examples available.

Carefully remove the board from the shipping carton. Check first if there are any damages before power on the evaluation board.

## *2.2 Features*



- socket adapter for H8/Tiny Modules (TiMo)
- In-Circuit serial Flash programming
- All resources available for evaluation
- All pins routed to connectors
- UART interface with MAX232 level converter and SubD-9 (female) connector
- Reset button
- 1 switch user/prog(programming)
- wire wrap field for user applications
- 5V/3V3 voltage regulator on board (switchable)

## 2.3 Board overview

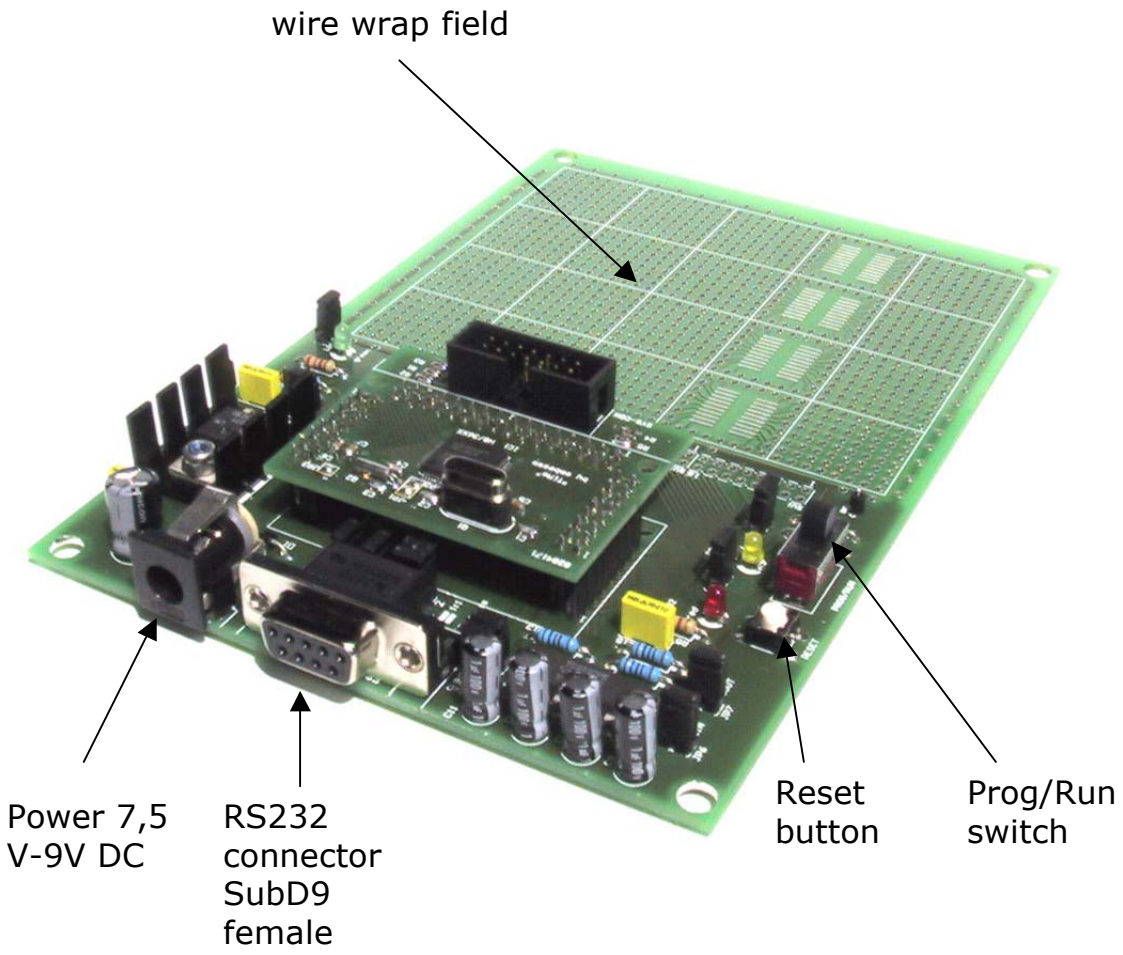



Figure 3 Board overview (TiBBo with TiMo)

## 2.4 Jumpers and switches

JP1 is used for switching the serial interface from 1:1 to crossed connection. If you connect a 1:1 cable (like the cable that comes with the optional kit) use the default setting. If you connect a crossed cable set the jumper to alternate setting.

| JP1 | Operation |
|---|---|
| 1-3, 2-4 | 1:1 operation |
| 1-2, 3-4 | Crossed connection |

JP2 is used to connect Vcc to the board. If the jumper is closed (1-2) Vcc is connected to all components on board.

| JP2 | Operation |
|---|---|
| Closed (default) | Vcc ON |
| Open | Vcc Off |

JP3 is for voltage selection. If you use the Vcc = 3V3 change the V24 level converter to a 3V3 type!

| JP3 | Operation |
|---|---|
| 1-2 | Vcc = 5V |
| 2-3 | Vcc = 3V3 |

JP4 selects the PowerOnReset (POR).

| JP4 | Operation |
|---|---|
| Closed (default) | POR on board |
| Open | POR open |

JP5 connects NMI to PullUp resistor and to Run/Prog switch

| JP5 | Operation |
|---|---|
| Closed (default) | NMI pulled up<br>Run/Prog switch enable |
| Open | NMI is open<br>No Run/Prog switch functionality |

JP6/JP7 containing signals to V24 level converter. Use them in dedicated functionality

| JP6 | Operation |
|---|---|
| 1 | T2 Output to V24 |
| 2 | R2 Input from V24 |

| JP7 | Operation |
|---|---|
| 1 | T2 Tx-Input |
| 2 | R2 Rx-Output |

JP8/JP9 connecting Tx/Rx signals from level converter to microcontroller.

| JP8 | Operation |
|---|---|
| Closed (default) | TxD enabled |
| Open | TxD disabled |

| JP9 | Operation |
|---|---|
| Closed (default) | RxD enabled |
| Open | RxD disabled |

JP10/JP11 are for direct connects on SubD9 connector.

| JP10 | Operation |
|---|---|
| Closed (default) | X1.4(DTR) and X1.6(DSR) connected |
| Open | |

| JP11 | Operation |
|---|---|
| Closed (default) | X1.7(RTS) and X1.8(CTS) connected |
| Open | |

PROG/RUN switches between RUN- and PROG(PROGRAMMING) mode. In programming mode the YELLOW LED is lighting.

| Orientation | Operation |
|---|---|
| LEFT | PROG(programming) mode |
| RIGHT | RUN mode |

RESET is for controller reset. While RESET the RED LED is lighting.

## *2.5 Connectors*

SubD_Female, Serial communication, SubD-9 female
PS-CON, Power connector

| PS-CON, Pin | Operation |
|---|---|
| 1 | GND |
| 2 | DC power supply, 7,5 – 9 VDC |

| SubD9, Pin | Operation | Remark |
|---|---|---|
| 1 | n.c. | Not connected |
| 2 | TXD | Connected to JP8 |
| 3 | RXD | Connected to JP9 |
| 4 | DTR | Connected to JP10 |
| 5 | GND | |
| 6 | DSR | Connected to JP10 |
| 7 | RTS | Connected to JP11 |
| 8 | CTS | Connected to JP11 |
| 9 | n.c. | Not connected |

SV1, SV2, SV3, SV4 connectors with microcontroller signals

| SV1 | | SV2 | | SV3 | | SV4 | |
|---|---|---|---|---|---|---|---|
| PIN | Operation | PIN | Operation | PIN | Operation | PIN | Operation |
| 1 | PB6 | 1 | P36 | 1 | P62 | 1 | P71 |
| 2 | PB7 | 2 | P37 | 2 | P61 | 2 | P72 |
| 3 | AVCC | 3 | P52 | 3 | /NMI | 3 | P14 |
| 4 | X2 | 4 | P53 | 3 | P60 | 4 | P15 |
| 5 | X1 | 5 | P54 | 5 | P64 | 5 | P16 |
| 6 | VCL | 6 | P55 | 6 | P65 | 6 | P17 |
| 7 | /RES | 7 | P10 | 7 | P66 | 7 | P33 |
| 8 | TEST(GND) | 8 | P11 | 8 | P67 | 8 | P32 |
| 9 | GND | 9 | P12 | 9 | P85 | 9 | P31 |
| 10 | OSC2 | 10 | P56 | 10 | P86 | 10 | P30 |
| 11 | OSC2 | 11 | P57 | 11 | P87 | 11 | PB3 |
| 12 | VCC | 12 | P74 | 12 | P20 | 12 | PB2 |
| 13 | P50 | 13 | P75 | 13 | P21 | 13 | PB1 |
| 14 | P51 | 14 | P76 | 14 | P22 | 14 | PB0 |
| 15 | P34 | 15 | P24 | 15 | P23 | 15 | PB4 |
| 16 | P35 | 16 | P63 | 16 | P70 | 16 | PB5 |

## *2.6 Start-Up instructions*

### 2.6.1 Installing the TiBBo

Installing the TiBBo requires a power supply and a serial connection to a host computer (common PC). The serial communications cable for connecting the TiBBo to a host computer is supplied and has 1:1 connectivity.

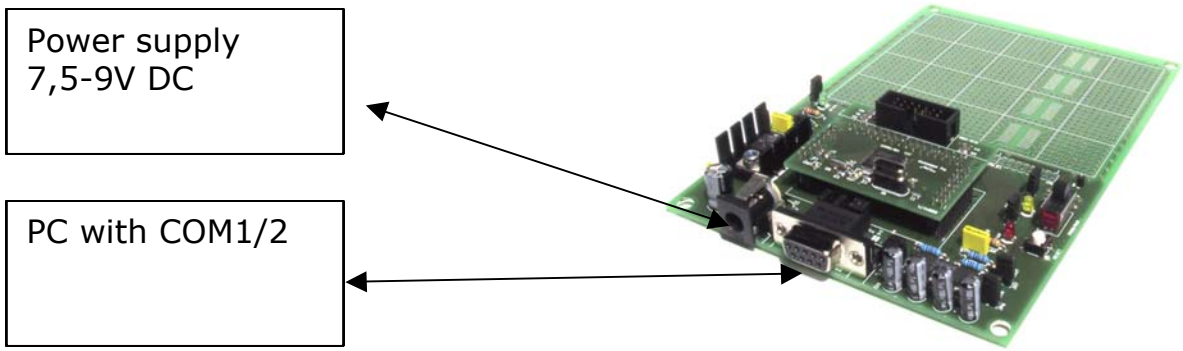



Figure 4 shows how to connect the TiBBo to a PC and to a power supply

### 2.6.2 Power Supply

The TiBBo hardware requires a power supply of 7,5V DC at minimum. Please don't use a power supply with more the 9V DC because the on board voltage regulator becomes very hot!

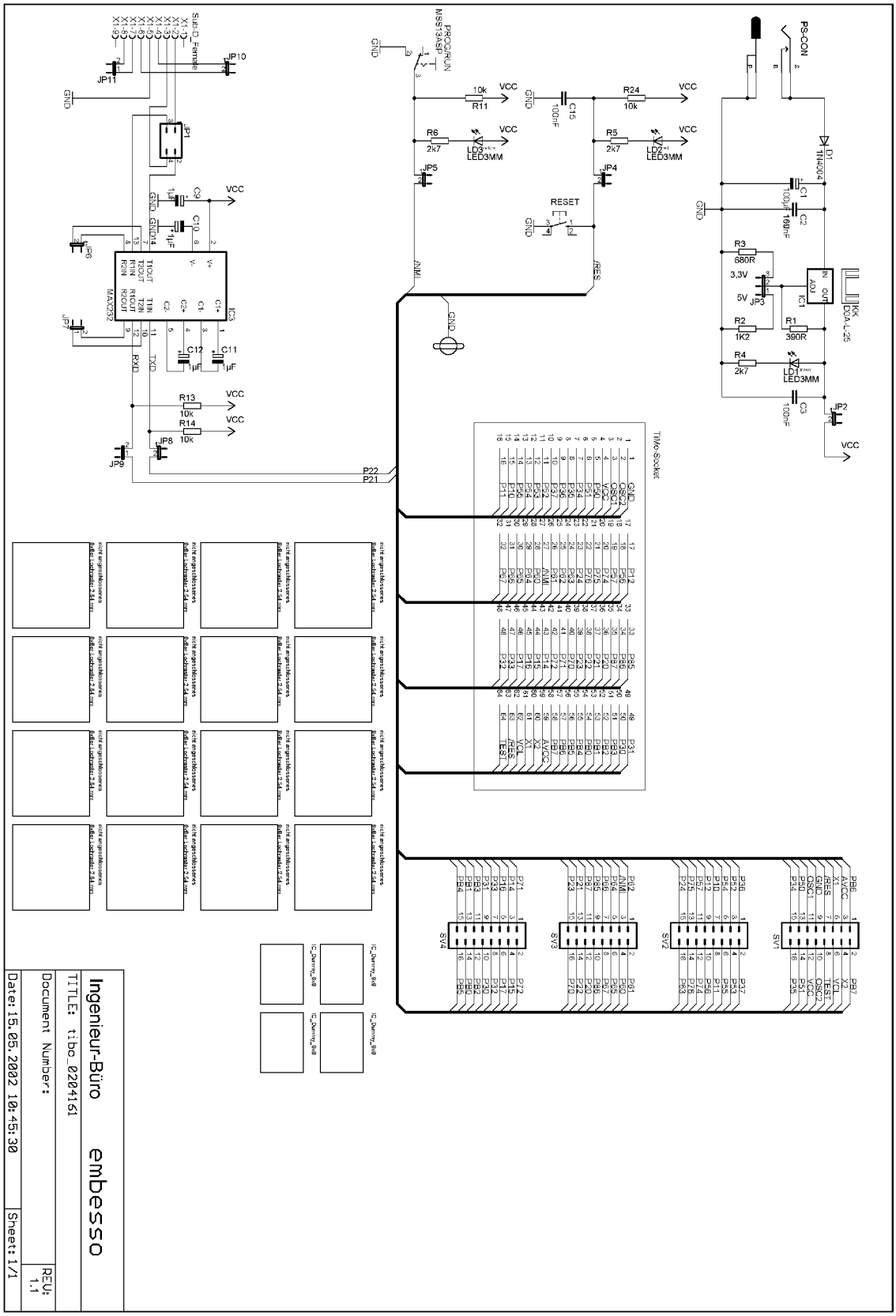
The power consumption is about 50mA without any user periphery. Since total power consumption can vary widely due to external connectors, H8/36XX port state, use a power supply capable of providing at least 300mA at +7,5V DC.

The design includes circuitry for reversed polarity protection.

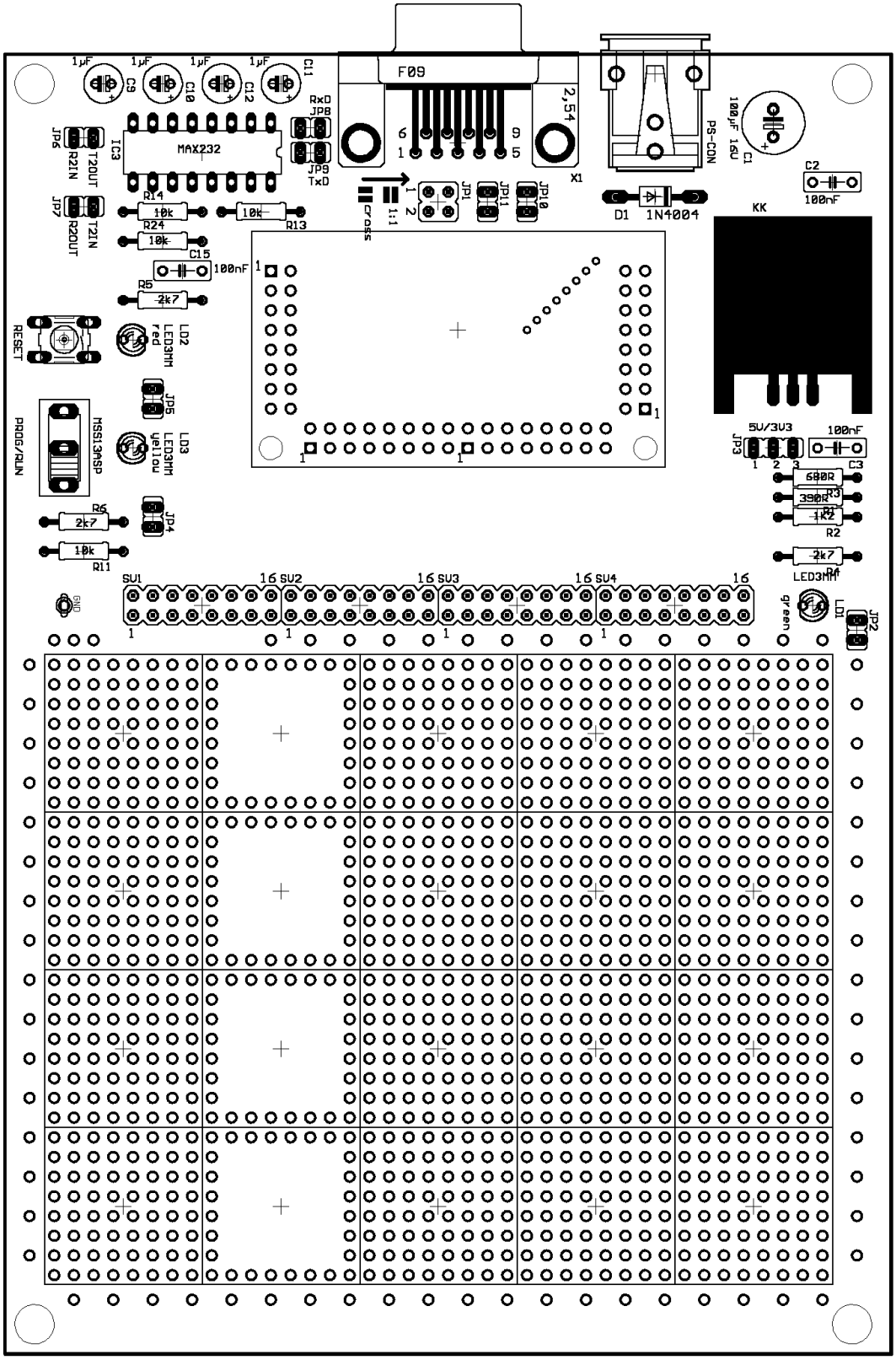
Please watch on GND (ground) connection between power supply, eval-board and PC.

After connecting to a power supply and power on the board the GREEN LED will lighting.

## *2.7 Schematics*



## 2.8 Board layout



# 3 Development Environment

## *3.1 Software Tools*

Software development on embesso-TiMo/TiBBo requires some software tools to be installed on your PC. All tools can be found on CD-R. Some of them must be installed separately. Please refer on installation / setup requirements.

You will find the following tools:

**EWH8:** IAR Embedded Workbench with a limited version of the IAR C compiler for all Hitachi Tiny controllers, assembler, linker and library generator

**HEW:** Hitachi Embedded Workbench with a limited version of the IAR C compiler for all Hitachi Tiny controllers, assembler, linker and library generator

**FDT:** A powerful freeware flash tool (flash-writer) from HMSE

Installation hints:

EWH8: Install EWH8 by start \programs\iar\autorun.exe . Follow the instructions in setup and look at the readme.txt file.

HEW: First install HEW with \programs\hew\setup.exe . In 3664files.zip you will find more definition files. Follow the instructions in setup and look at the readme.txt file.

FDT: Next install \programs\fdt\ftd15.exe. Then the plugin fdt3664f.exe must be installed. Follow the setup instructions. A documentation will be found in fdt_man.pdf.

If you have installed these tools please refer to the next lessons for workflow.

**NOTE: Most freeware tools are unsupported versions! Please refer to manuals or hints on website for FAQ's!**

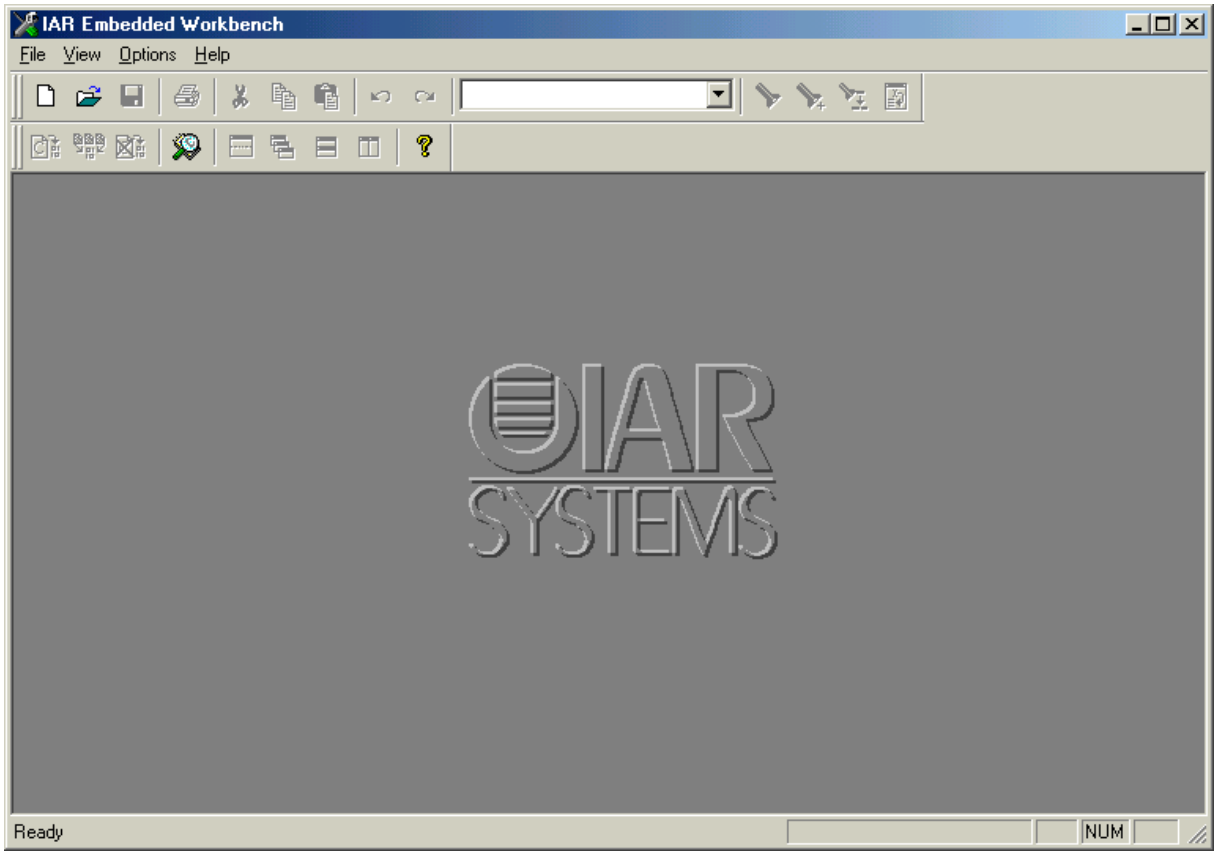
It is strongly recommended to refer all additional documents like H8/3664F hardware manual and H8 programming manual. Please see the application notes and several readme files on CD-R. Sometimes you should watch on the Hitachi, HMSE and IAR websites for tool upgrading, news and latest versions of all tools.

Hitachi: www.hitachi-eu.com/semiconductors
HMSE: www.hmse.com
IAR: www.iar.com

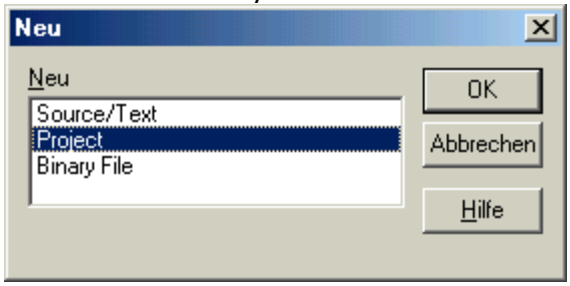
## *3.2 Create a program using IAR-EWH8*

Software development can be done with a integrated embedded workbench like IAR-EWH8. This software contains an editor, some tools for organization and a tool chain for compiling, assembling and linking programs.

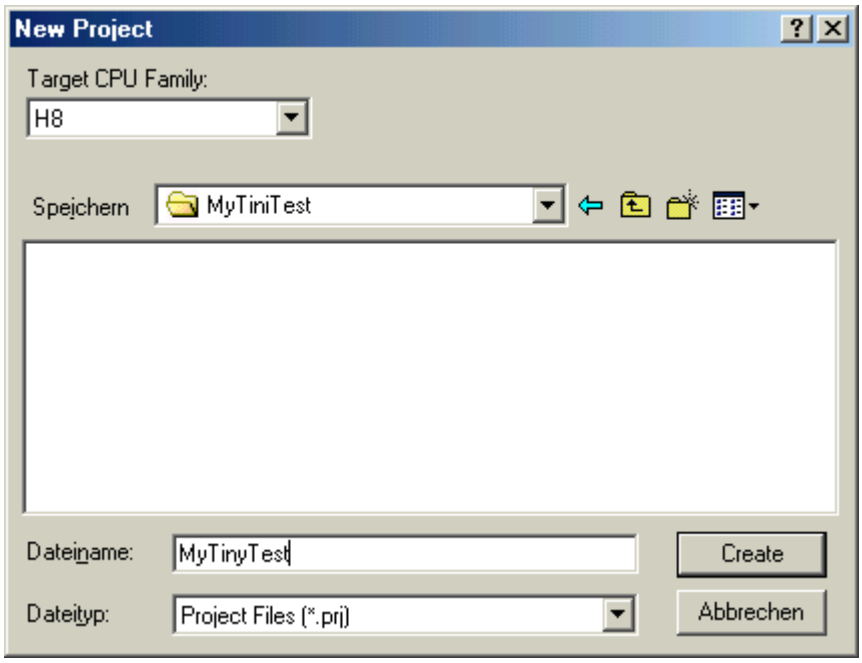
Start IAR Embedded Workbench on your PC. The following window will appear:



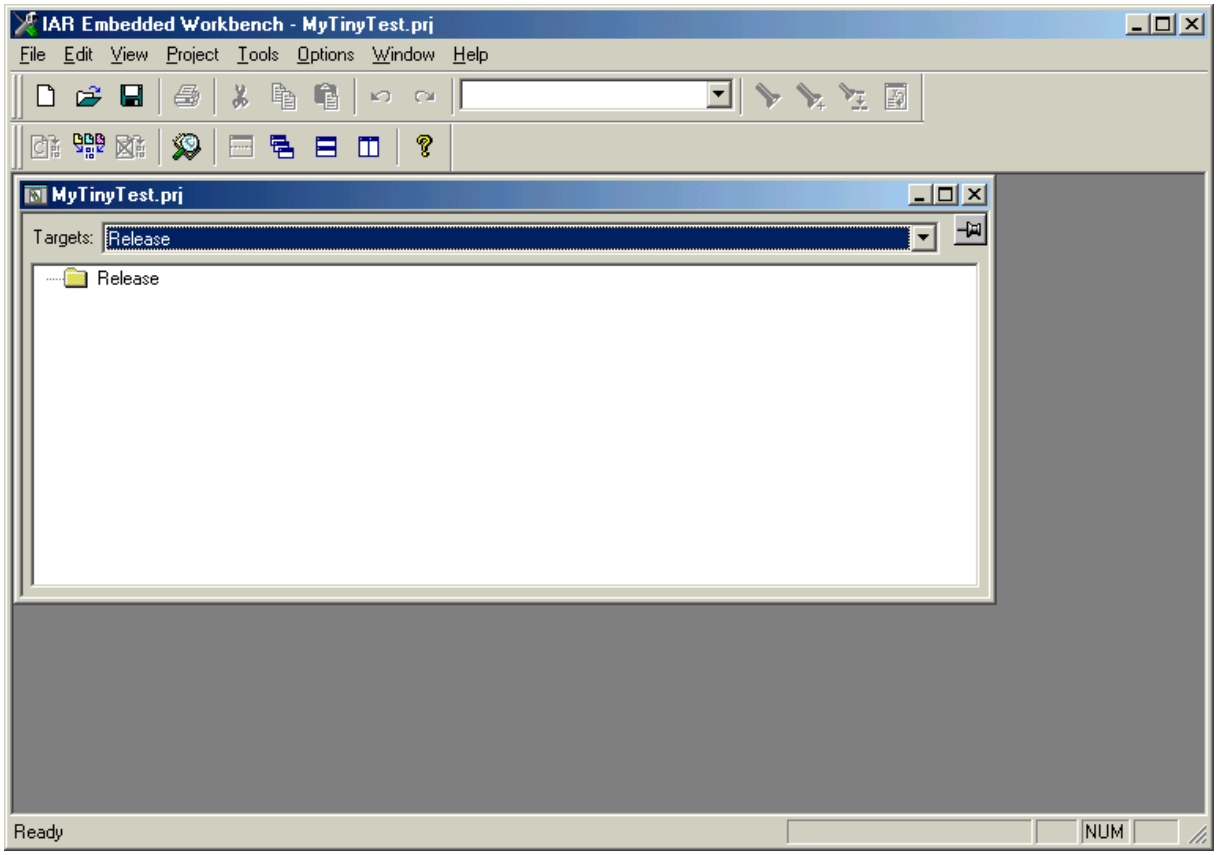

Now select File / New and select “Project”



Press OK and a file window will appear. Here first create a new directory (e.g. c:\MyTinyTest) and type the project filename “MyTinyTest”. After that click CREATE.



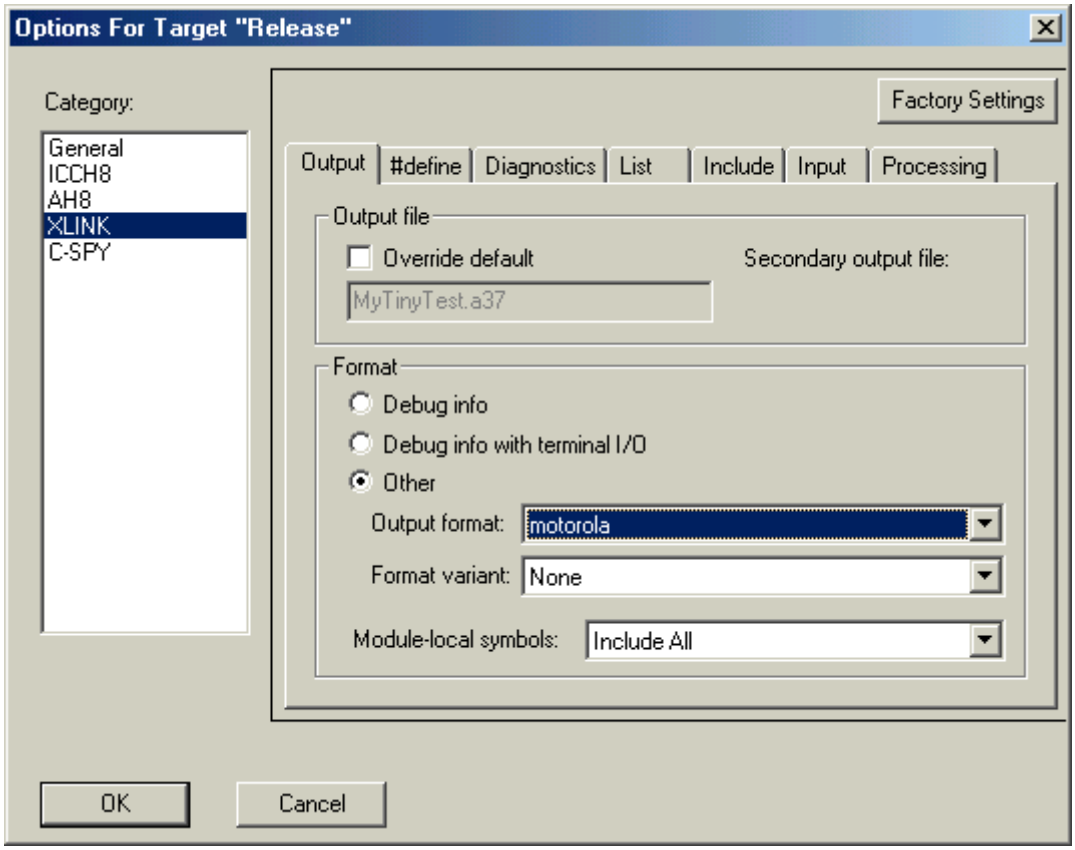

Now a new project is created and we must do some settings. In the window select under targets: “RELEASE”.



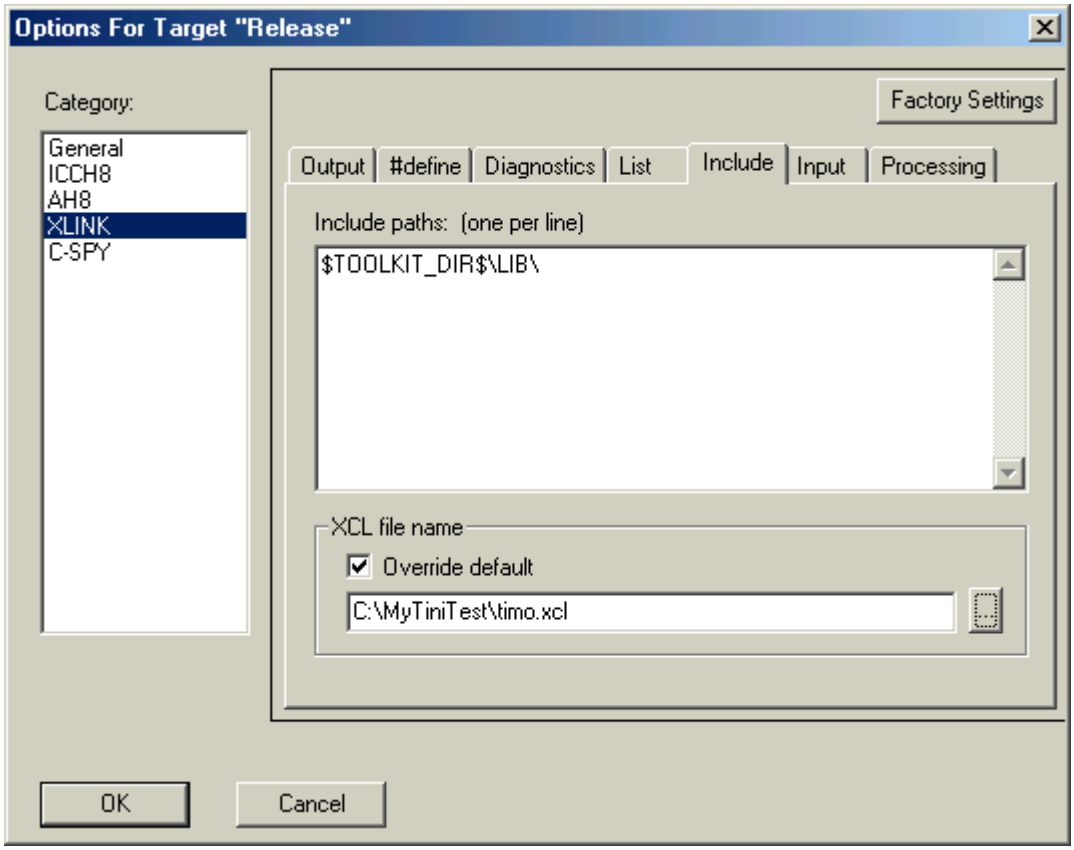

Now select release with the right mouse button. A popup appears. Select Options… and do the following settings:

In selection ICCH8/List select the List file box.

In section XLINK/Output select under Format “motorola” as the output format.




On the CD-R you will find a file called "timo.xcl". That file must be used as the xlink input file. Please copy it to your target directory and select in section Input/XCL file name the file timo.xcl.

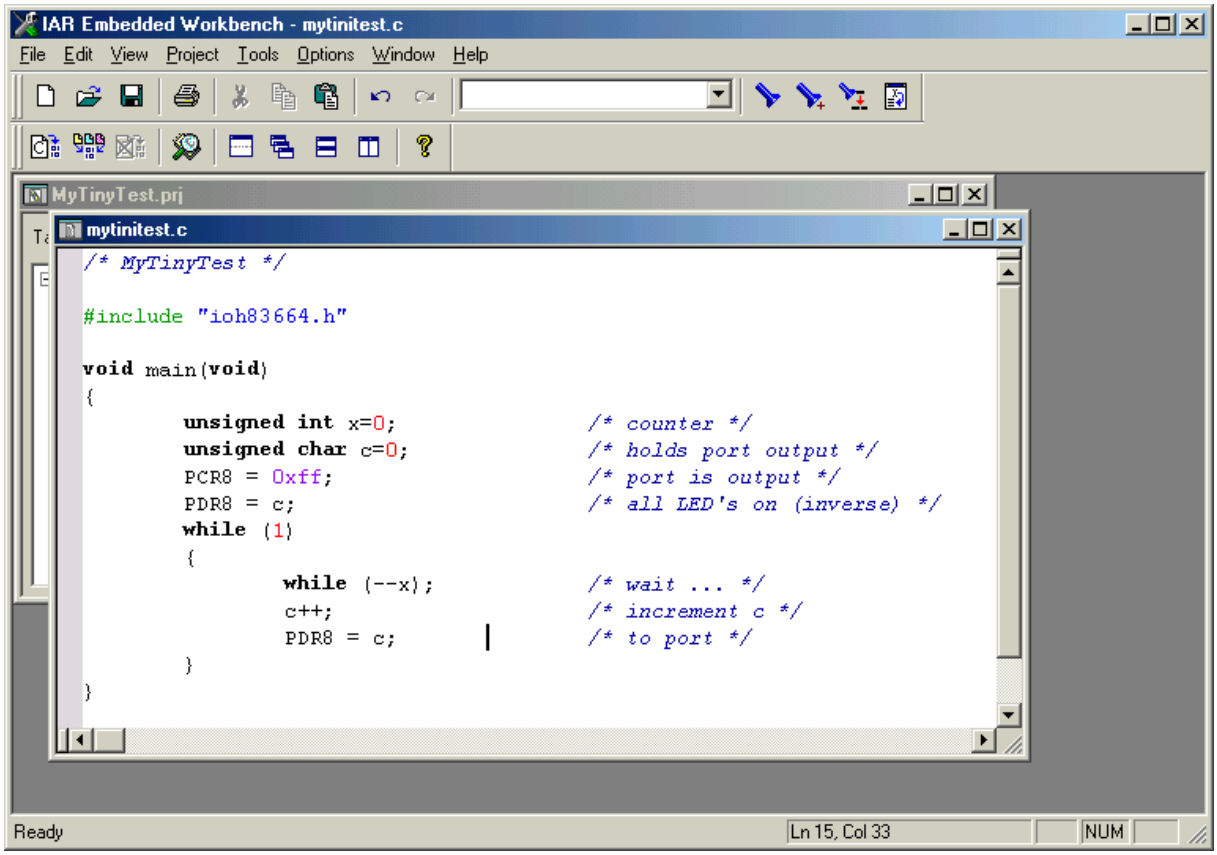



All other options can be changed later.

Click on OK.

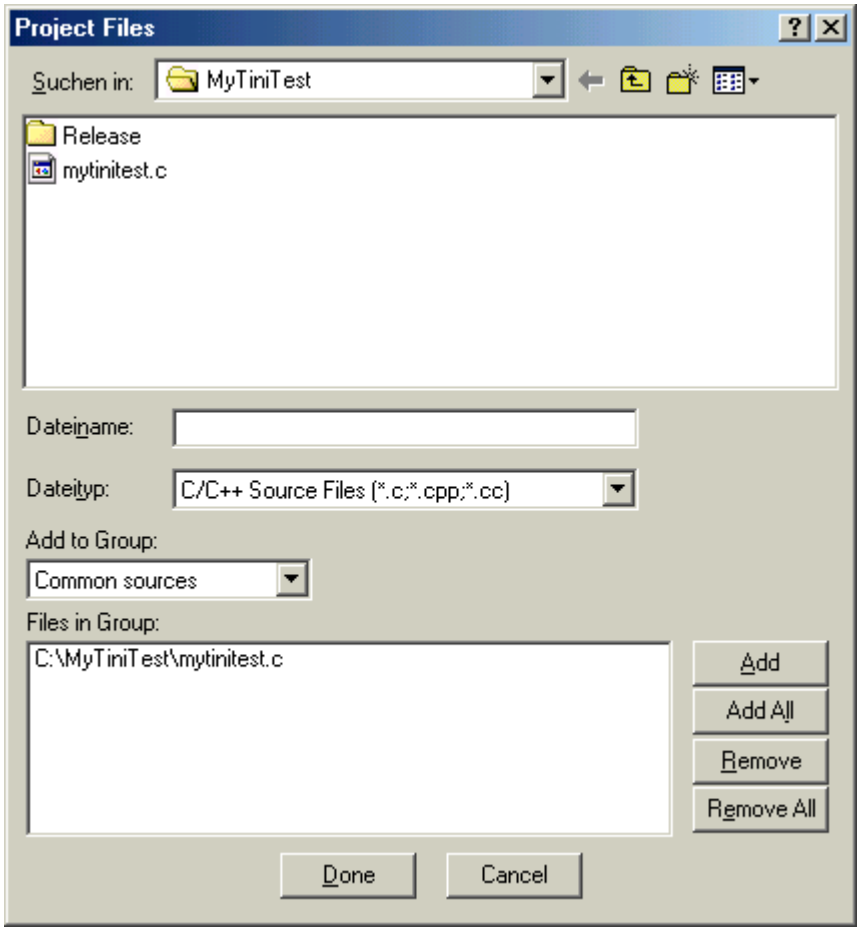
Now select File/new/source file and type in the following program:

```
/* MyTinyTest */

#include "ioh83664.h"

void main(void)
{
    unsigned int x=0;                /* counter */
    unsigned char c=0;               /* holds port output */
    PCR8 = 0xff;                     /* port is output */
    PDR8 = c;                        /* all LED's on (inverse) */
    while (1)
    {
        while (--x);                 /* wait ... */
        c++;                         /* increment c */
        PDR8 = c;                    /* to port */
    }
}
```



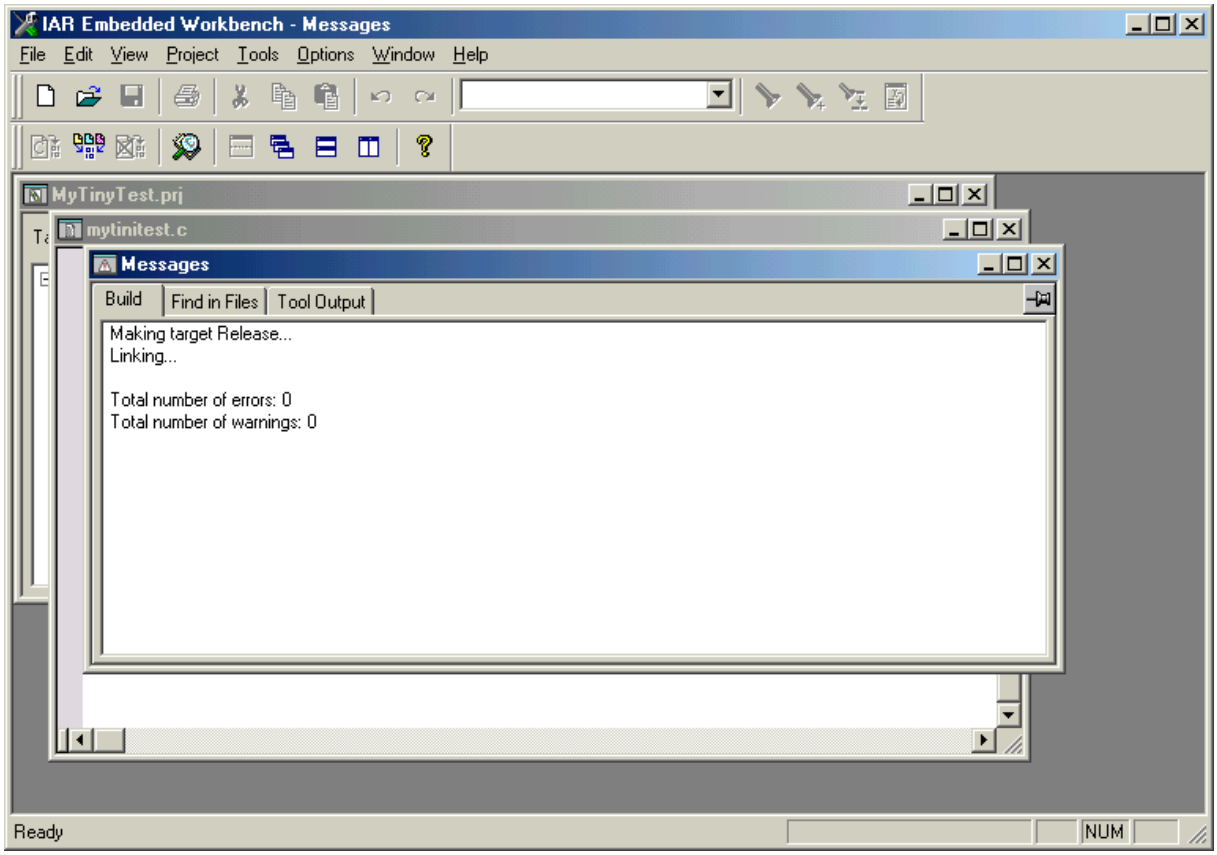

After that save it under MyTinyTest.c

Now we must add this file to our project. Please select Project/Files and add the file MyTinyTest.c.




After that click on DONE.

Now you can select Projet/Build ALL (or F9) and all files are compiled and linked. The message window shows if your project is error free or if there are any errors.




The target file for download can be found in directory c:\mytinytest\release\exe\mytinytest.a37.

Please see chapter “FDT” for information about downloading this file to target system.

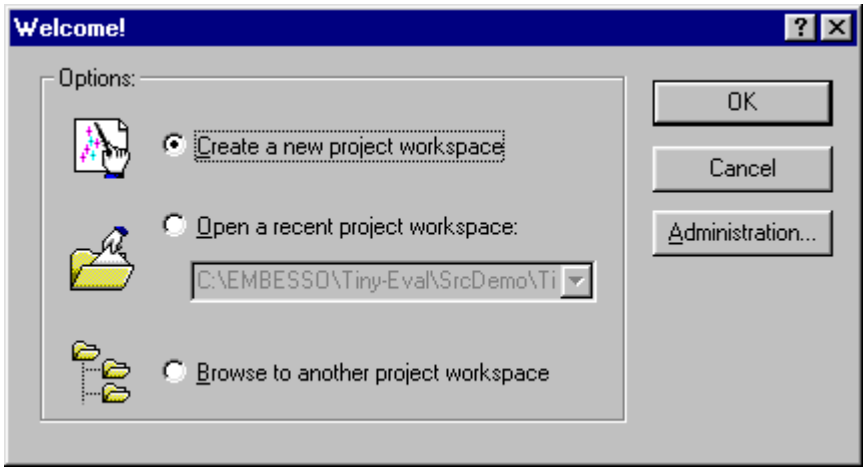
## *3.3 Create a program using HEW*

Software development can be done with the integrated embedded workbench like HEW. This software contains an editor, some tools for organization and a tool chain for compiling, assembling and linking programs.

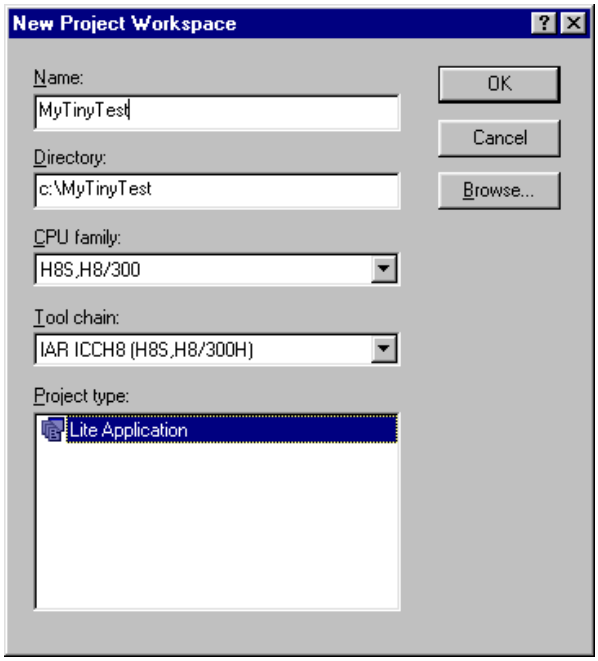
There is a version of the HEW with IAR tool chain for H8/Tiny microcontrollers from HMSE included.

For creating a new project please follow the next Steps:

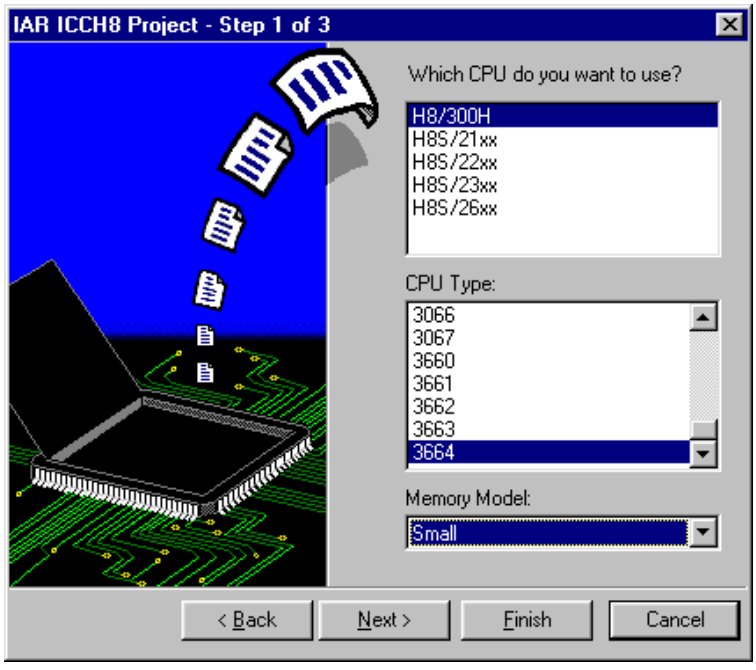
Start HEW and create a new workspaces:




select "create a new project workspace", click „OK" and type in the (project-) name.

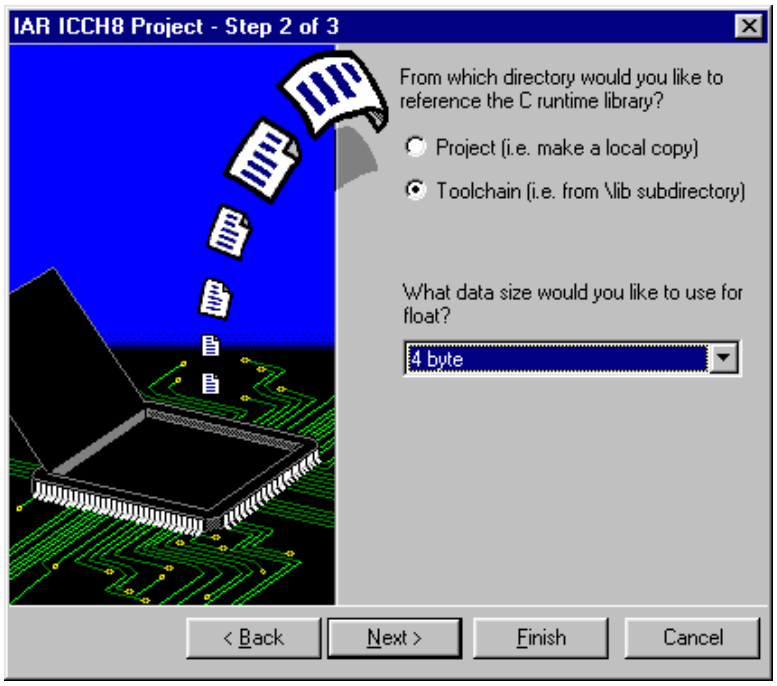



Here the project name is "MyTinyTest". A project directory will be added automatically but can be changed manually – please use the defaults here. CPU-family is H8S/H8/300 (H8/300H is the Tiny series). Click "OK" when ready.

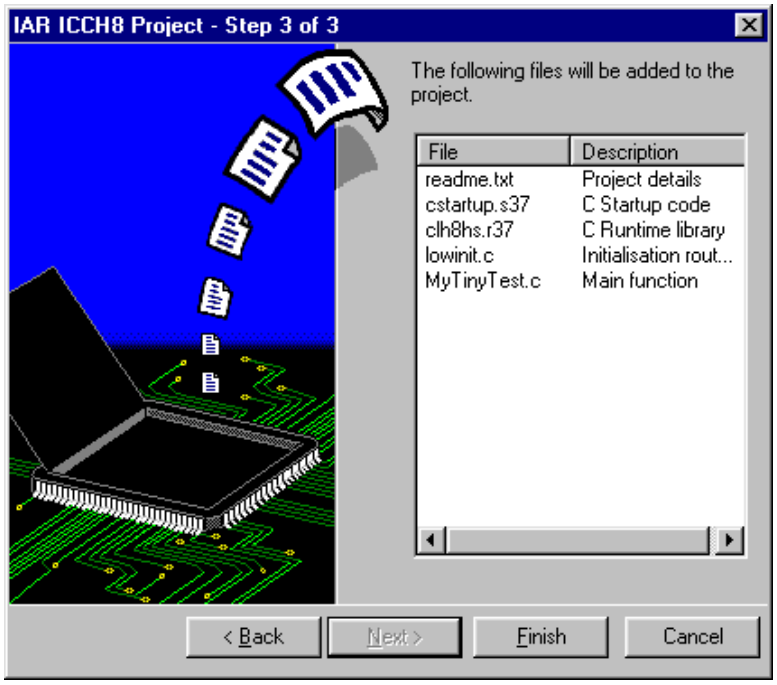



Next select CPU type = „H8/300H". CPU is 3664. The Memory model should be small because the Tiny series contains at least 56kByte address area.

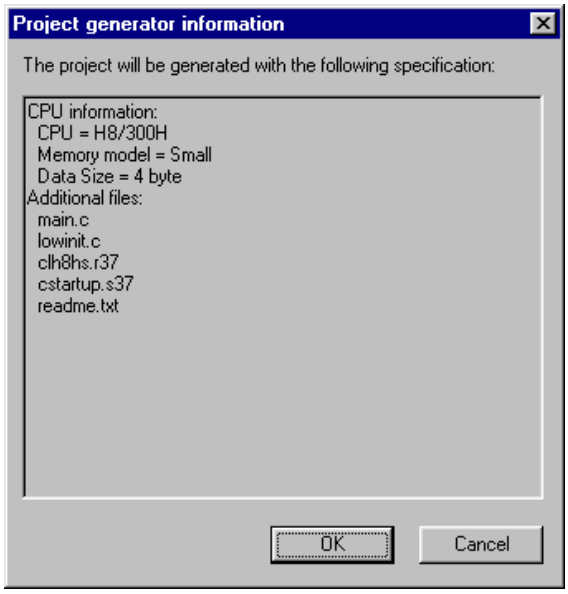
Then click ”next“.

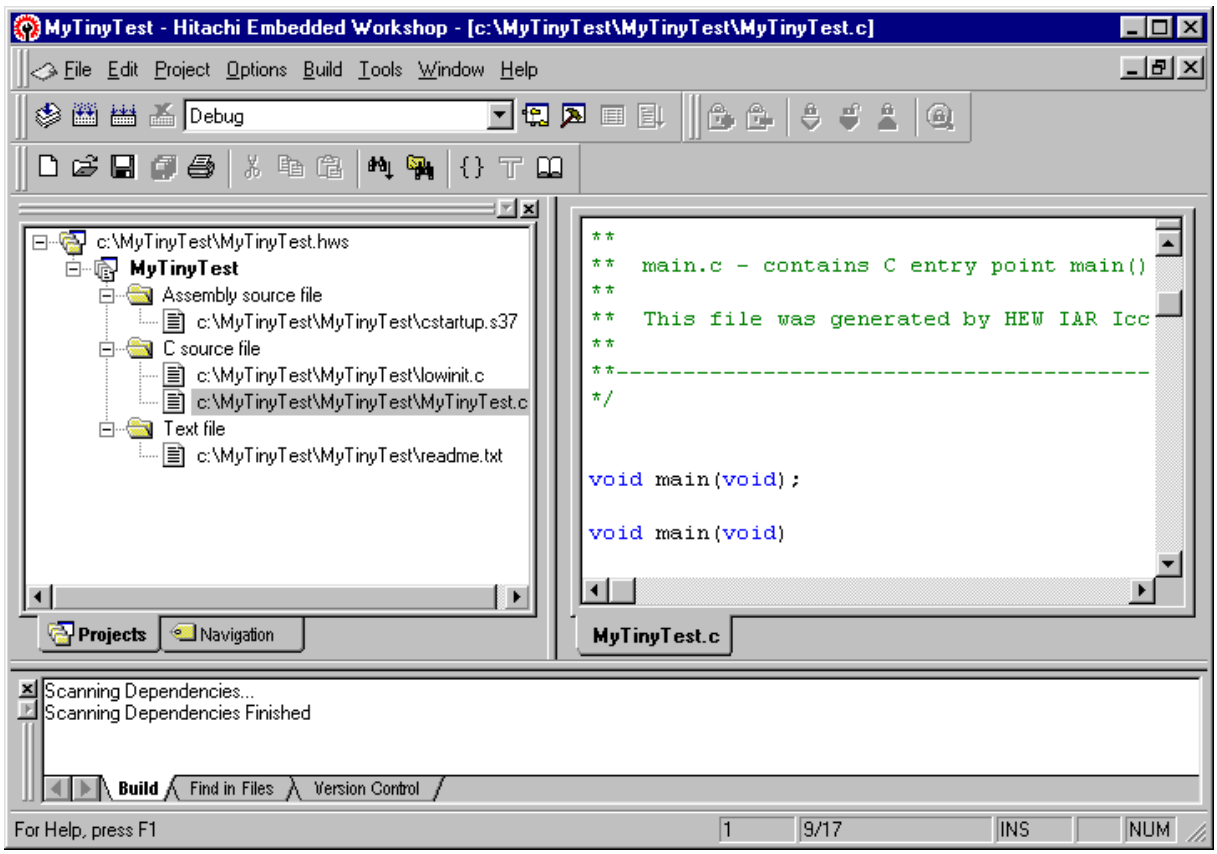

IAR ICCH8 Project - Step 2 of 3
From which directory would you like to reference the C runtime library?
Project (i.e. make a local copy)
Toolchain (i.e. from \lib subdirectory)
What data size would you like to use for float?
4 byte
< Back
Next >
Finish
Cancel

Projekt-Runtime-Files can be in a local or toolchain directory. Then you have to decide, if for your project float (4 Byte) or double (8Byte) floating point is necessary.




Now you can check your inputs and go back if anything must be changed. Click on “Finish” closes the wizard.



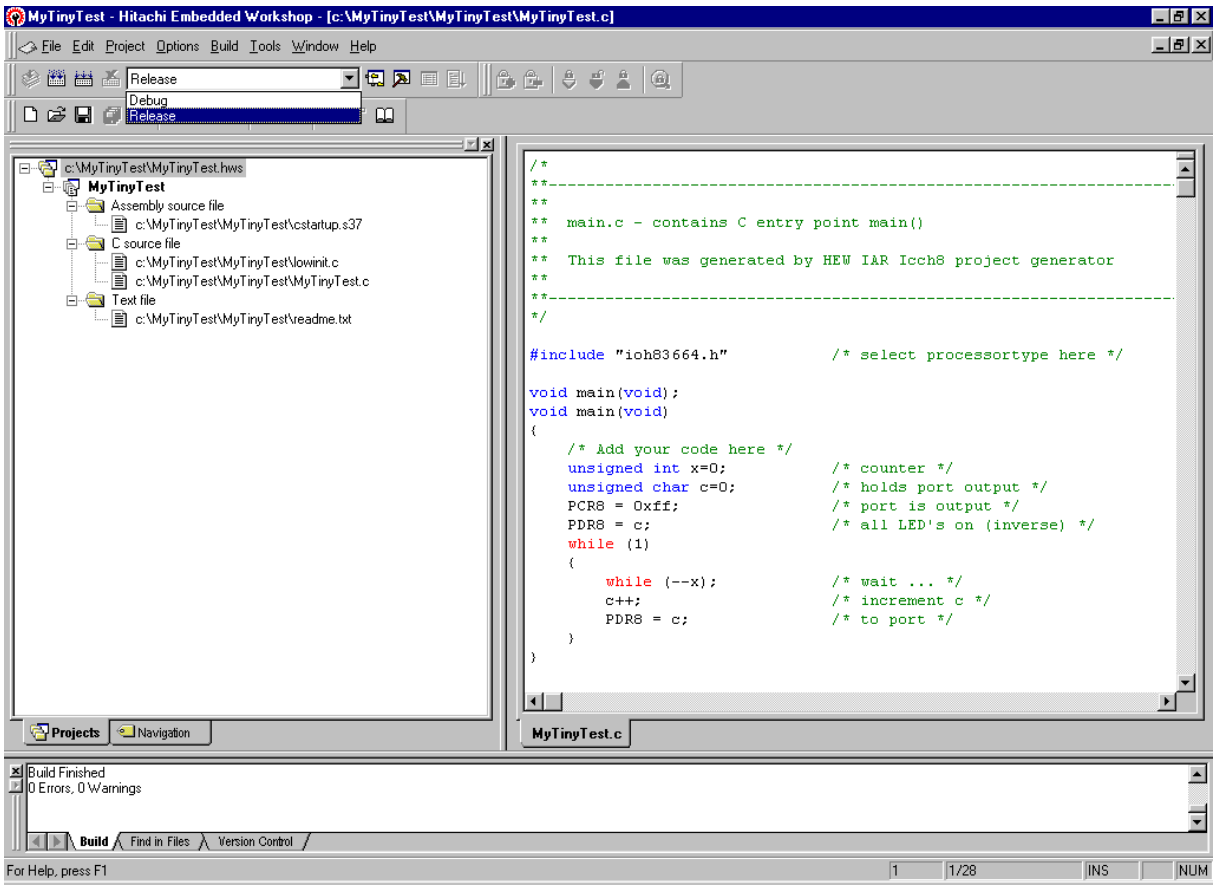




Now your project is well prepared. At the left side you can see the structure of your project with all files. A double click on 'C source file' / …\MyTinyTest.c will open the file to the right window.

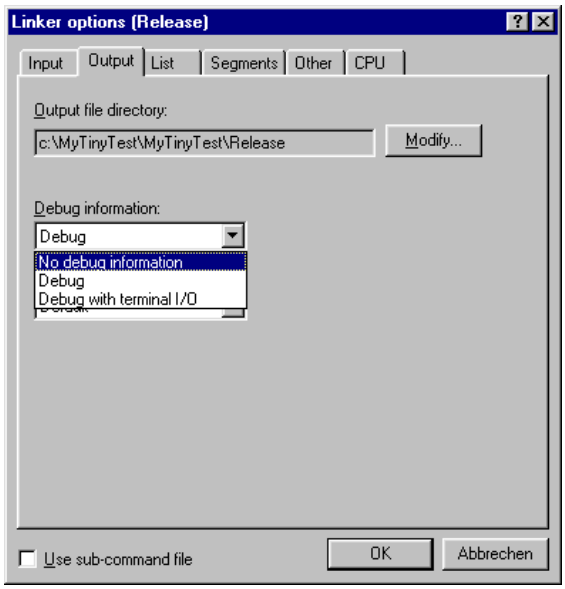
Please save your project with file/save.

Now type in the following program:

```
/*
**-------------------------------------------------------------------------
**
**  main.c - contains C entry point main()
**
**   This file was generated by HEW IAR Icch8 project generator
**
**-------------------------------------------------------------------------
*/

#include "ioh83664.h"                /* select processortype here */

void main(void);
void main(void)
{
    /* Add your code here */
    unsigned int x=0;                /* counter */
    unsigned char c=0;               /* holds port output */
    PCR8 = 0xff;                     /* port is output */
    PDR8 = c;                        /* all pin's on (inverse) */
    while (1)
    {
        while (--x);                 /* wait ... */
        c++;                         /* increment c */
        PDR8 = c;                    /* to port */
    }
}
```



Now select "release" and change the linker option (release) 'Debug information' to "No debug information".



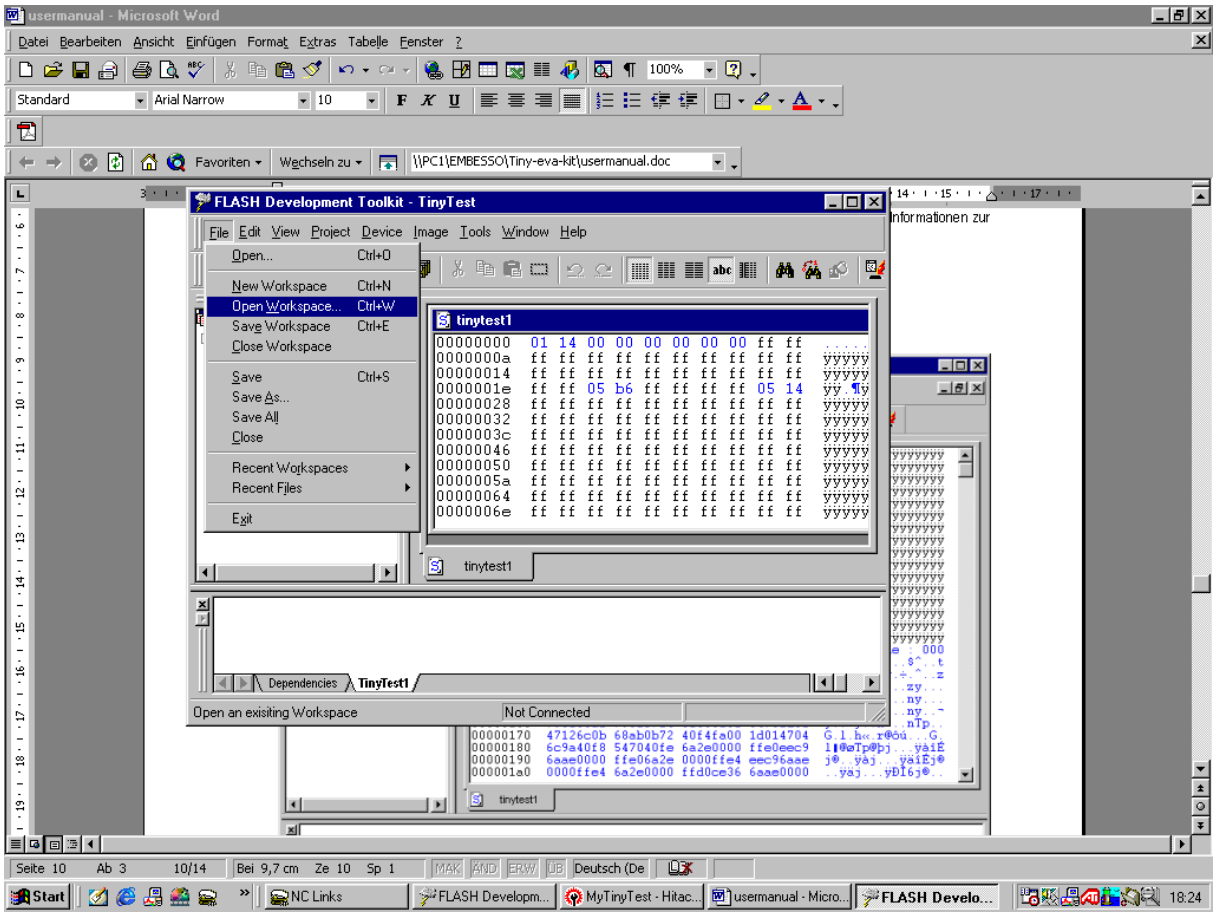

With "Build All" (or F7 = Build) the project will be compiled and linked. If there are no errors the message window shows:

```
Building All - MyTinyTest - Release
Phase Compiler starting
c:\MyTinyTest\MyTinyTest\MyTinyTest.c
c:\MyTinyTest\MyTinyTest\lowinit.c
Phase Compiler finished

Phase Assembler starting
c:\MyTinyTest\MyTinyTest\cstartup.s37
Phase Assembler finished

Phase Linker starting
Phase Linker finished

Build Finished
0 Errors, 0 Warnings
```

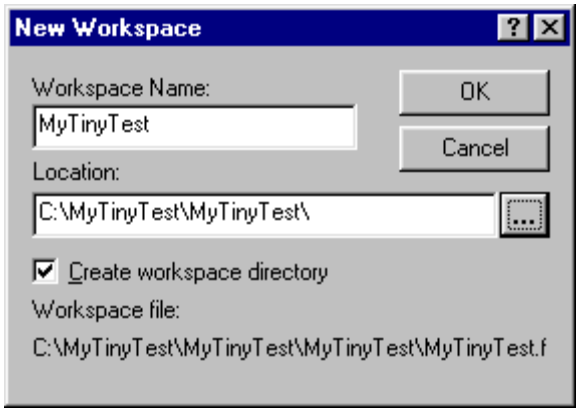
## *3.4 Download the code using FDT*

After compiling and linking (error free!), the target code (mytinytest.d37) should be downloaded to target board. Therefore we use a freeware tool from HMSE : **FDT**.

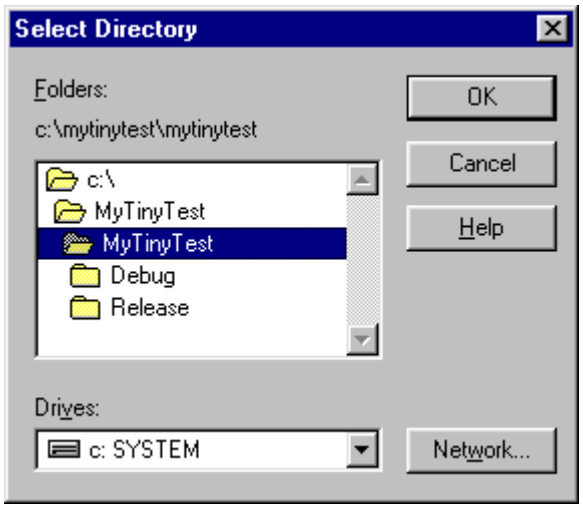
Even FDT must be prepared for a new workspace.



Please start FDT and select „New Workspace". Here we use the project name "MyTinyTest".

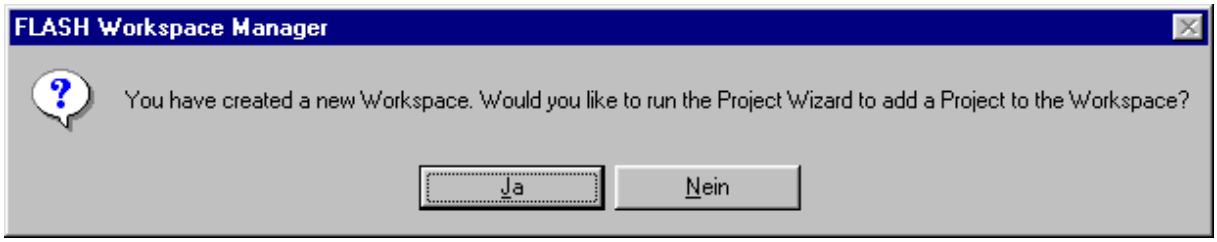



You can choose a location for all workspace files. Select on subdirectory from „MyTinyTest".



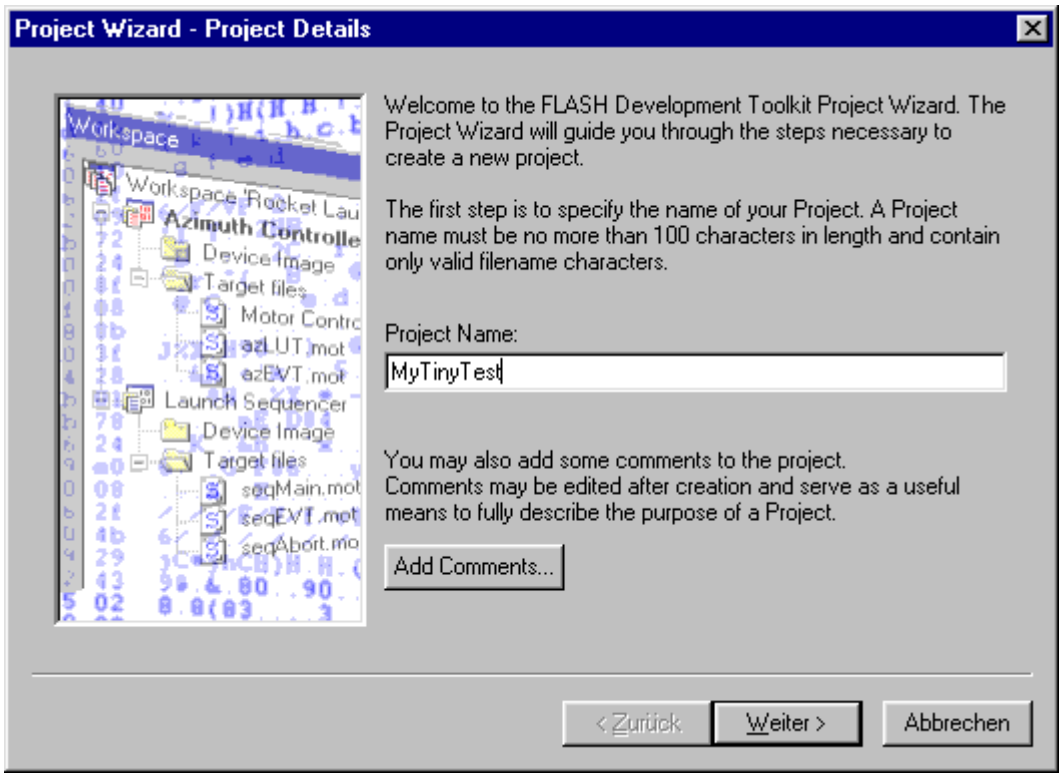

Click ok and a further window will appear:

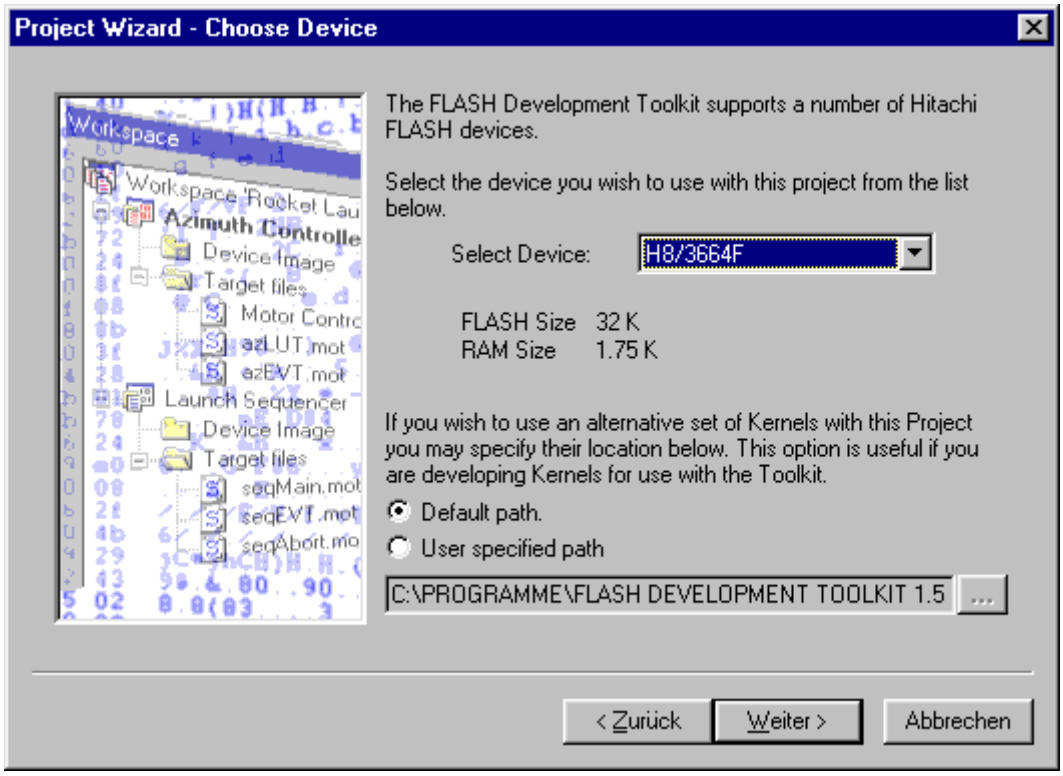
Select „Yes"

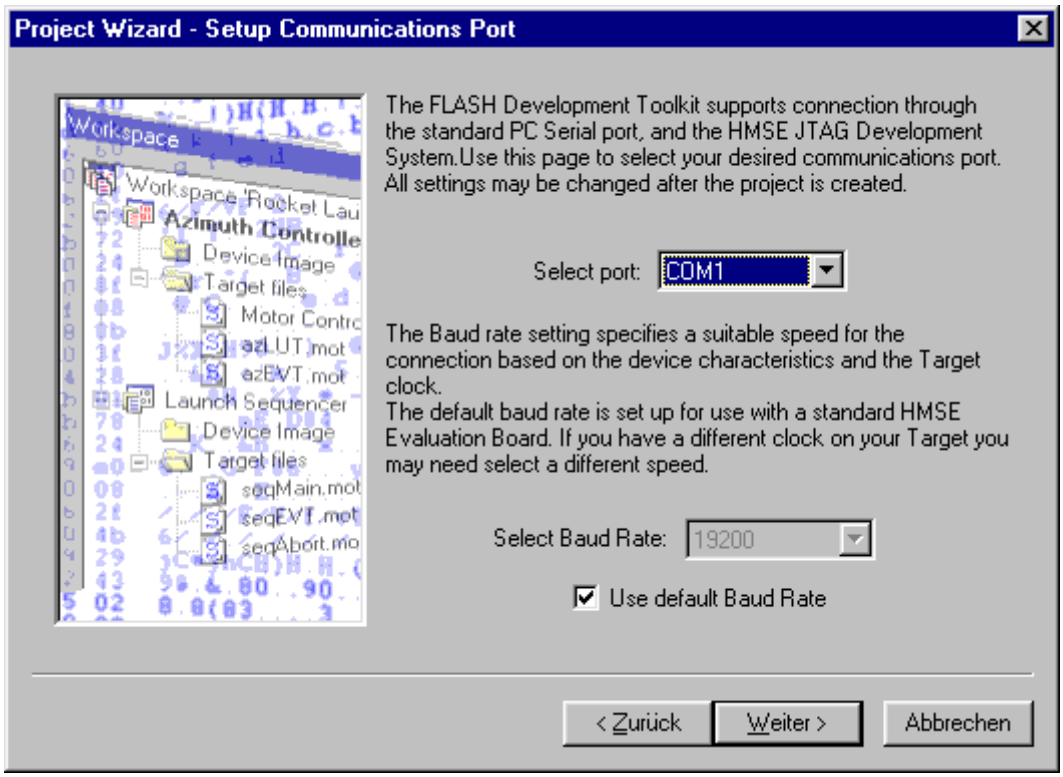

FLASH Workspace Manager
You have created a new Workspace. Would you like to run the Project Wizard to add a Project to the Workspace?
Ja
Nein

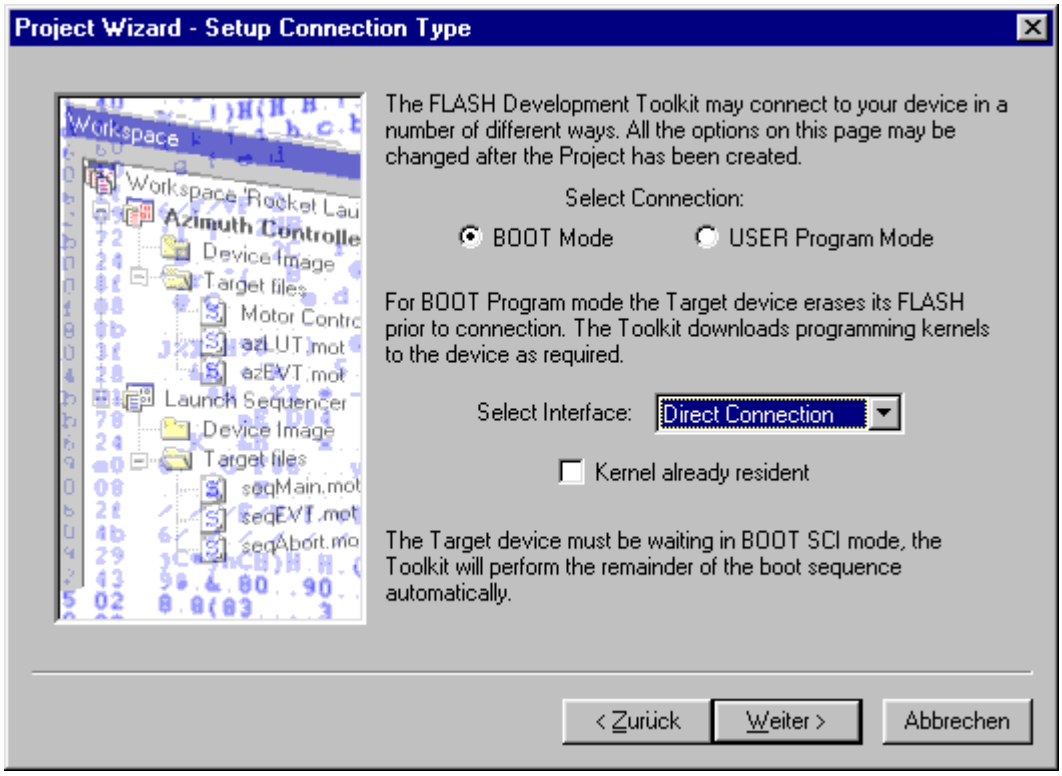
First time users should use the wizard!

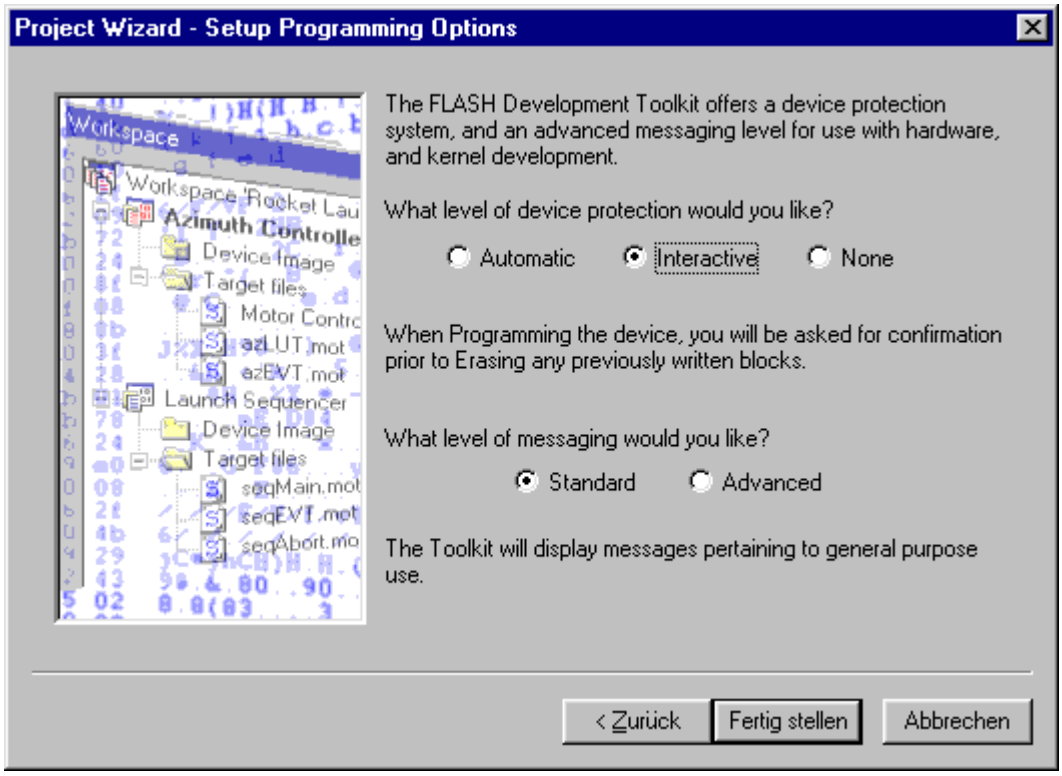
Fill in the following things:

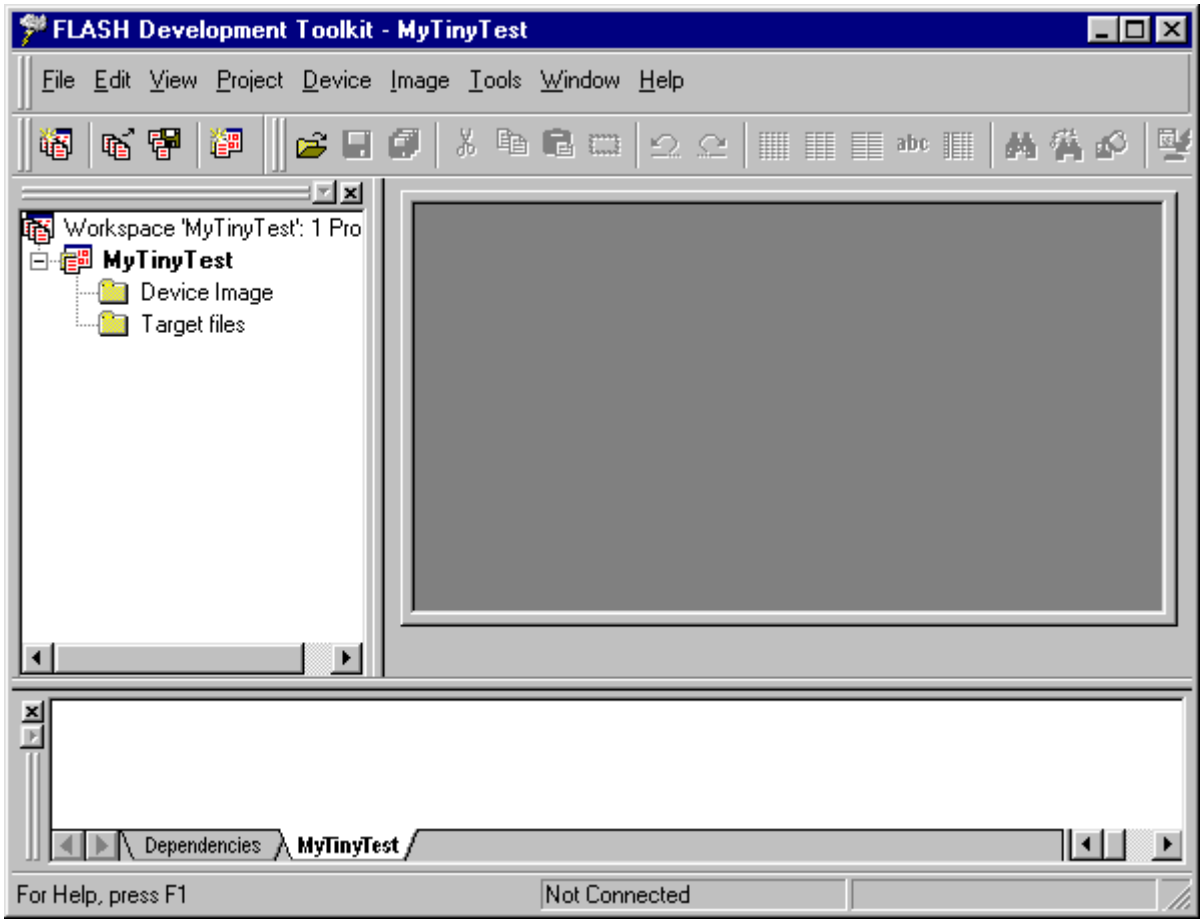

Project Wizard - Project Details
Workspace
Workspace 'Rocket Lau
Azimuth Controlle
Device Image
Target files
Motor Contro
azLUT.mot
azEVT.mot
Launch Sequencer
Device Image
Target files
seqMain.mot
seqEVT.mot
seqAbort.mo
Welcome to the FLASH Development Toolkit Project Wizard. The Project Wizard will guide you through the steps necessary to create a new project.
The first step is to specify the name of your Project. A Project name must be no more than 100 characters in length and contain only valid filename characters.
Project Name:
MyTinyTest
You may also add some comments to the project. Comments may be edited after creation and serve as a useful means to fully describe the purpose of a Project.
Add Comments...
< Zurück
Weiter >
Abbrechen


Project Wizard - Choose Device
Workspace
Workspace 'Rocket Lau
Azimuth Controlle
Device Image
Target files
Motor Contro
azLUT.mot
azEVT.mot
Launch Sequencer
Device Image
Target files
seqMain.mot
seqEVT.mot
seqAbort.mo
The FLASH Development Toolkit supports a number of Hitachi FLASH devices.
Select the device you wish to use with this project from the list below.
Select Device:
H8/3664F
FLASH Size 32 K
RAM Size 1.75 K
If you wish to use an alternative set of Kernels with this Project you may specify their location below. This option is useful if you are developing Kernels for use with the Toolkit.
Default path.
User specified path
C:\PROGRAMME\FLASH DEVELOPMENT TOOLKIT 1.5
...
< Zurück
Weiter >
Abbrechen


Project Wizard - Setup Communications Port
Workspace
Workspace 'Rocket Lau
Azimuth Controlle
Device Image
Target files
Motor Contro
azLUT.mot
azEVT.mot
Launch Sequencer
Device Image
Target files
seqMain.mot
seqEVT.mot
seqAbort.mo
The FLASH Development Toolkit supports connection through the standard PC Serial port, and the HMSE JTAG Development System.Use this page to select your desired communications port. All settings may be changed after the project is created.
Select port:
COM1
The Baud rate setting specifies a suitable speed for the connection based on the device characteristics and the Target clock.
The default baud rate is set up for use with a standard HMSE Evaluation Board. If you have a different clock on your Target you may need select a different speed.
Select Baud Rate:
19200
Use default Baud Rate
< Zurück
Weiter >
Abbrechen

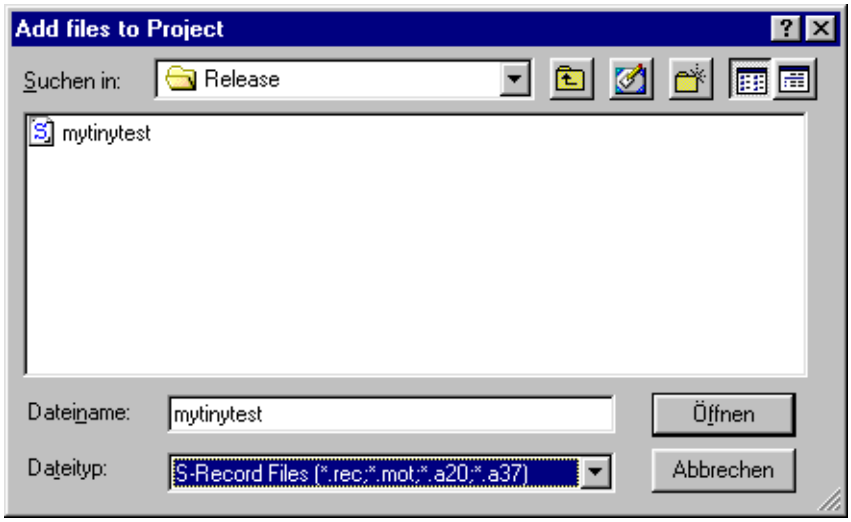

Project Wizard - Setup Connection Type
Workspace
Workspace 'Rocket Lau
Azimuth Controlle
Device Image
Target files
Motor Contrc
azLUT.mot
azEVT.mot
Launch Sequencer
Device Image
Target files
seqMain.mot
seqEVT.mot
seqAbort.mo
The FLASH Development Toolkit may connect to your device in a number of different ways. All the options on this page may be changed after the Project has been created.
Select Connection:
BOOT Mode
USER Program Mode
For BOOT Program mode the Target device erases its FLASH prior to connection. The Toolkit downloads programming kernels to the device as required.
Select Interface:
Direct Connection
Kernel already resident
The Target device must be waiting in BOOT SCI mode, the Toolkit will perform the remainder of the boot sequence automatically.
< Zurück
Weiter >
Abbrechen


Project Wizard - Setup Programming Options
Workspace
Workspace 'Rocket Lau
Azimuth Controlle
Device Image
Target files
Motor Contrc
azLUT.mot
azEVT.mot
Launch Sequencer
Device Image
Target files
seqMain.mot
seqEVT.mot
seqAbort.mo
The FLASH Development Toolkit offers a device protection system, and an advanced messaging level for use with hardware, and kernel development.
What level of device protection would you like?
Automatic
Interactive
None
When Programming the device, you will be asked for confirmation prior to Erasing any previously written blocks.
What level of messaging would you like?
Standard
Advanced
The Toolkit will display messages pertaining to general purpose use.
< Zurück
Fertig stellen
Abbrechen

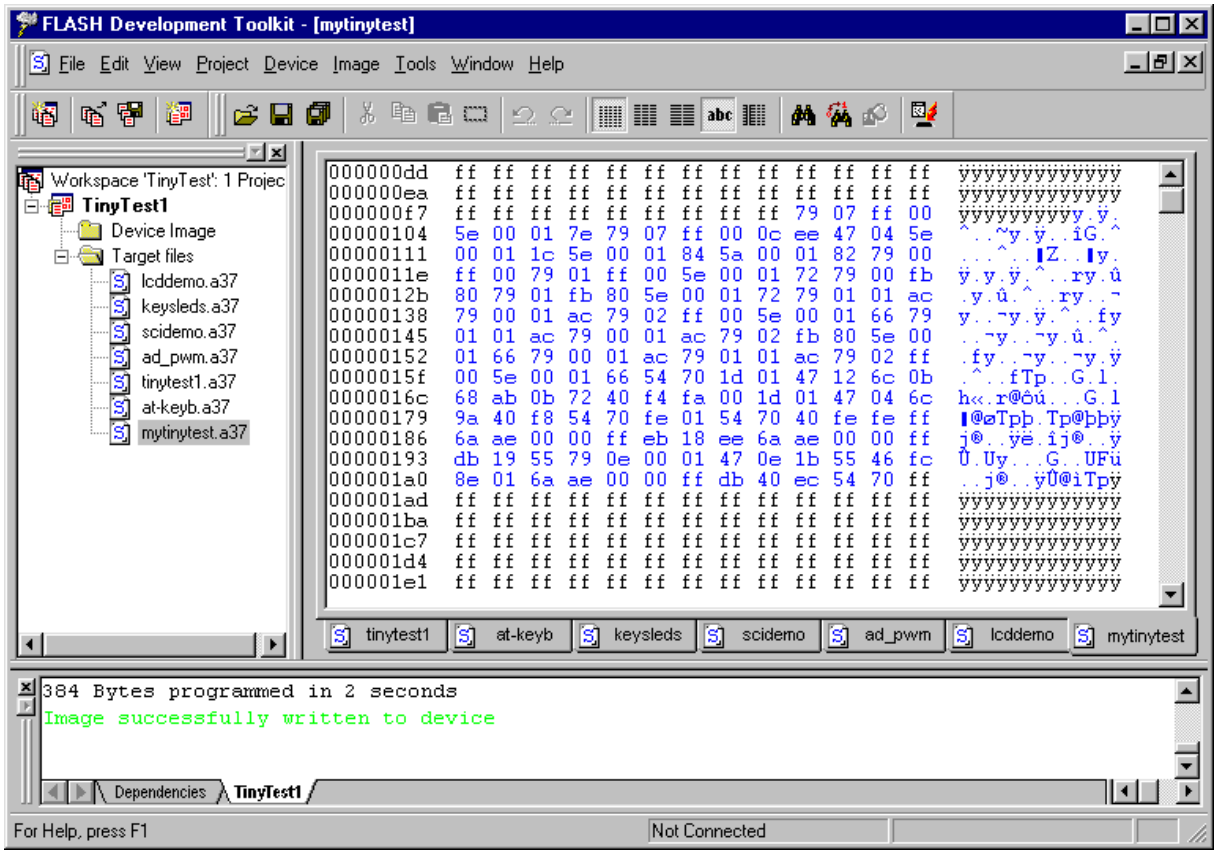



Now a workspace is created and you can add your target file to „TargetFiles":

Select Project/Add new files to project… and search for file: c:\mytinytest\mytinytest\release\mytinytest.mot.



Now make a double click on \targetfiles\mytinytest and the file content of mytinytest.mot will appear in hex format in the right window.


FLASH Development Toolkit - [mytinytest]
File Edit View Project Device Image Tools Window Help
Workspace 'TinyTest': 1 Projec
TinyTest1
Device Image
Target files
lcddemo.a37
keysleds.a37
scidemo.a37
ad_pwm.a37
tinytest1.a37
at-keyb.a37
mytinytest.a37
000000dd ff ff ff ff ff ff ff ff ff ff ff ff ff
000000ea ff ff ff ff ff ff ff ff ff ff ff ff ff
000000f7 ff ff ff ff ff ff ff ff ff 79 07 ff 00
00000104 5e 00 01 7e 79 07 ff 00 0c ee 47 04 5e
00000111 00 01 1c 5e 00 01 84 5a 00 01 82 79 00
0000011e ff 00 79 01 ff 00 5e 00 01 72 79 00 fb
0000012b 80 79 01 fb 80 5e 00 01 72 79 01 01 ac
00000138 79 00 01 ac 79 02 ff 00 5e 00 01 66 79
00000145 01 01 ac 79 00 01 ac 79 02 fb 80 5e 00
00000152 01 66 79 00 01 ac 79 01 01 ac 79 02 ff
0000015f 00 5e 00 01 66 54 70 1d 01 47 12 6c 0b
0000016c 68 ab 0b 72 40 f4 fa 00 1d 01 47 04 6c
00000179 9a 40 f8 54 70 fe 01 54 70 40 fe fe ff
00000186 6a ae 00 00 ff eb 18 ee 6a ae 00 00 ff
00000193 db 19 55 79 0e 00 01 47 0e 1b 55 46 fc
000001a0 8e 01 6a ae 00 00 ff db 40 ec 54 70 ff
000001ad ff ff ff ff ff ff ff ff ff ff ff ff ff
000001ba ff ff ff ff ff ff ff ff ff ff ff ff ff
000001c7 ff ff ff ff ff ff ff ff ff ff ff ff ff
000001d4 ff ff ff ff ff ff ff ff ff ff ff ff ff
000001e1 ff ff ff ff ff ff ff ff ff ff ff ff ff
tinytest1
at-keyb
keysleds
scidemo
ad_pwm
lcddemo
mytinytest
384 Bytes programmed in 2 seconds
Image successfully written to device
Dependencies
TinyTest1
For Help, press F1
Not Connected

First press the reset button at the target board, hold it down and move Prog/Run-switch to prog position (left). After that release the reset button.

With Image/Download image (Ctrl-P) you one can start the connection setup to target board and start downloading image file.

Now press Ctrl-P (Download) on FDT and the download process will start. Watch on progress bar while download.

When the download is finished press Alt-C to disconnect the PC connection.

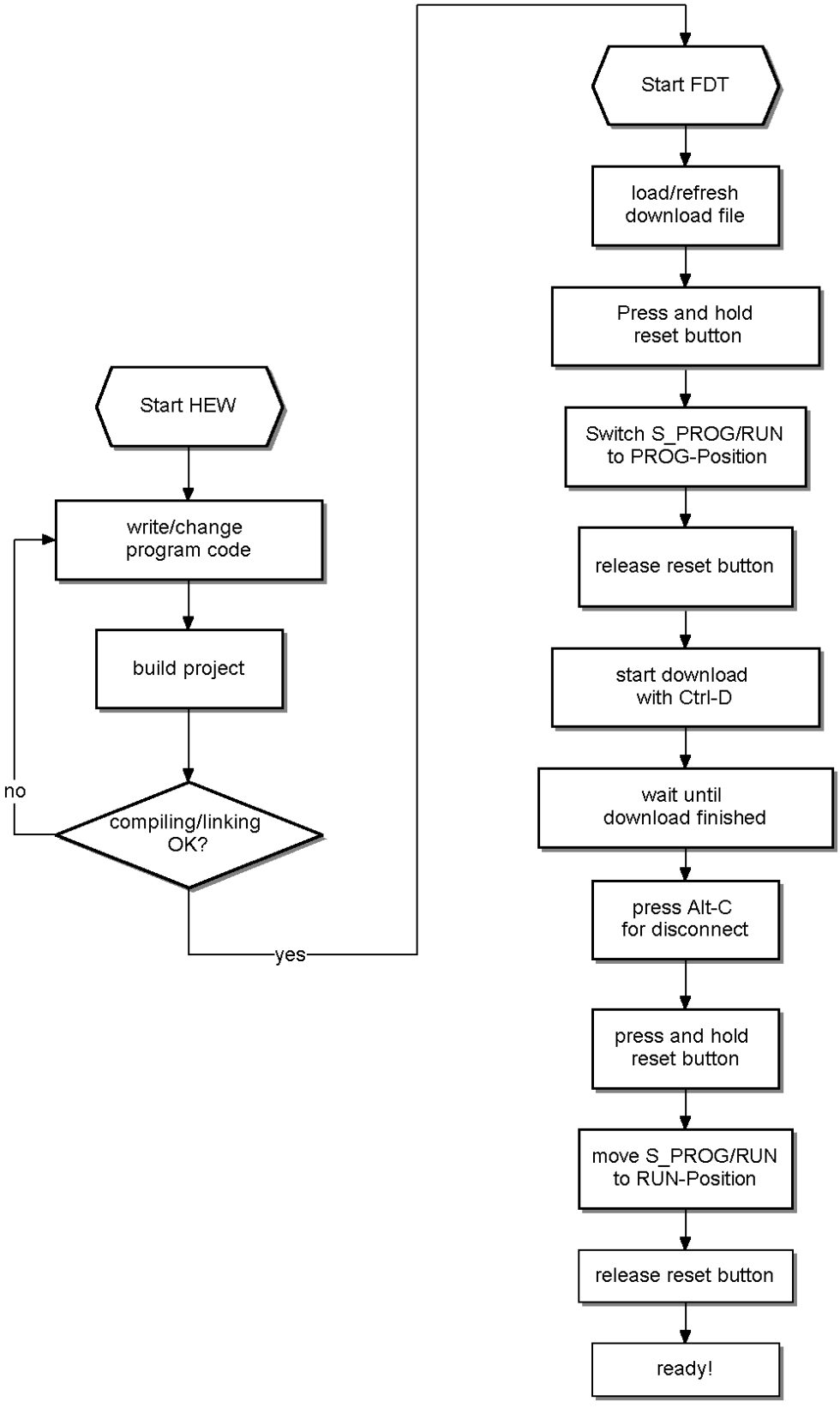
On target board, move Prog/Run-switch to run position (right) press down the reset button and release reset button.

Congratulations! Now your first program is running!

You will see the LED's flickering.

Now you can do some additional functions in HEW. After compiling and linking only go to FDT, update your download file with the command Freshen all Target files (Ctrl-T), reconnect the link and repeat the download process.

## 3.5 Workflow


Start HEW
write/change program code
build project
compiling/linking OK?
no
yes
Start FDT
load/refresh download file
Press and hold reset button
Switch S_PROG/RUN to PROG-Position
release reset button
start download with Ctrl-D
wait until download finished
press Alt-C for disconnect
press and hold reset button
move S_PROG/RUN to RUN-Position
release reset button
ready!

# 4 Examples

TiMo/TiBBo is provided with some demonstration code.

On the supplied CD-R you should find a complete prepared workspace for the different embedded workbenches (IAR-EWH8 and HEW) in the directory \examples\demoapp\ .

Please copy the complete directory to your hard disk in a directory "c:\embesso\TiMo\", so you will finally have the following directory "c:\embesso\TiMo\examples\demoapp\" with all application notes included.

Then start EWH8 or HEW and select "open existing workspace". Select one of the projects and do your exercises.

For all projects we need the same header file containing some definitions and the include file for the target microcontroller H8/3664F. So if you want to work with these files don't forget to include the file "mydef.s" first in your project file:

```
#ifndef _MYDEFS_H_
#define _MYDEFS_H_

#include "ioh83664.h"               // select processortype here
#include "inh8.h"
#include "icclbutl.h"

#define CPU_CLK  9830400            // select clk for diff. calc.

#ifndef NULL
#define NULL 0x00
#endif

#ifndef FALSE
#define FALSE 0x00
#endif

#ifndef TRUE
#define TRUE 0x01
#endif

typedef unsigned char u8;
typedef unsigned int u16;

#endif
```

## 4.1 Timer_A

The first demo program shows the usage of TIMER_A as a simple clock source – toggling one port bit. For time-controlling we use TIMER_A as an periodic interval timer. The interrupt service routine (isr) is toggling one port bit.

```
/*-------------------------------------------------------------------
**  Timer_A is used for timer tick with irq
**-------------------------------------------------------------------*/

#include "mydefs.h"                         // for all nec. includes

/***************************************
    TimerA-Interrupt (1s)
***************************************/
interrupt [TIMER_A] void Timer_A_Isr(void)
{
    // here toggle port bit
    IRR1 &= ~0x40;                          // clear irq-flag
}

/***************************************
    TimerA-Test
    setup : CLK/8 (=1µs @ 8MHz) at P10 (TMOW)
             1s-Irq-intervall @ SubClock (32.678 Hz)
***************************************/
void Timer_A_Init(void)
{
    PCR8 = 0xff;                            // P8 = output
    TMA = 0x0c;                             // Reset PrescalerW
    TMA = 0x4b;                             // CLK/8 on P10, 1/32s-interval (Clk=Prescaler W)
    PMR1 |= 0x01;                           // set TMOW (P10) = Output
    IENR1 |= 0x40;                          // enable TimerA-Interrupt
    set_interrupt_mask(0);           // enable all interrupts
}

void main(void)
{
    Timer_A_Init();                         // init & start timer_A
    while(1);                               // just wait ...
}
```

## *4.2 SCI*

SCI is used here for a simple RS232 (V24) terminal connection. Please use a terminal program like HyperTerm (included in Windows), select Baudrate 9600 Baud, 8 Databits, No Parity and 1 Stopbit (8N1). After connection and setup, hit some keys and you will see a message responding on every keycode sent.

```
/*
**---------------------------------------------------------------------------
** scidemo.c uses the sci for connection with a terminal program like
** windows HyperTerm. Setup = 9600,8,N,1
**---------------------------------------------------------------------------
*/

#include "mydefs.h"           // see file for further include

/* some defines */
#define TIE         0x80
#define RIE         0x40
#define TE          0x20
#define RE          0x10
#define MPIE        0x08
#define TEIE        0x04

#define CK_INT      0x00
#define CK_INT_OUT  0x01
#define CK_EXT      0x02

#define IS_SCI_RDF      (SSR & 0x40)
#define CLEAR_SCI_RDF   SSR = (SSR & ~0x40)

#define IS_SCI_TX_FREE  (SSR & 0x80)

#define V24_BRR(x)    ((unsigned char)(((CPU_CLK+16*x)/32/x) - 1))


void V24Init (u16 Baudrate)
{
 SCR3 = 0x00;                         // disable all
 SSR = 0x00;                          // clear all errorbits
 SMR = 0x00;                          // 8N1 + /1 clock
 BRR = V24_BRR(Baudrate);     // set baud
 PMR1 |= 0x02;                        // P22 = TxD Output
 SCR3 = (TE|RE|CK_INT);               // Ints und Data disabled, internal clock
}

u8 V24NewChar(void)                   // check for new char on V24
{
 if (IS_SCI_RDF)                      // Receive buffer full?
 {
  return TRUE;
 }
 return FALSE;
}

u8 V24GetChar(u8* data)               // simple GetChar via V24
{
  u8 idx;
 if (IS_SCI_RDF)                      // Receive buffer full?
 {
  *data = RDR;                        // yes, get data
  CLEAR_SCI_RDF;                      // clear RDRF-Bit
  return TRUE;
 }
 return FALSE;
```

```
}


u8 V24PutChar(u8 c)                  // simple PutChar via V24
{
  if (IS_SCI_TX_FREE)         // Tx register free ?
  {
    TDR = c;                          // yes, put data in tx register
    return TRUE;
  }
  return FALSE;
}


u8 V24Write(u8 *s)                   // simple Write(string) via V24
{
  while (*s != 0)                    // while not end of string
  {
    if (V24PutChar(*s) == TRUE) s++;  // PutChar
  }
  return TRUE;
}


u8 V24WriteLn(u8 *s)          // simple WriteLine (string + CR/LF)
{
  u8 ret = FALSE;
  ret  = V24Write(s);
  ret |= V24Write("\n\r");
  return ret;
}


void ShowUse(void)                   // simple menu
{
    V24WriteLn("\n\n\rV24-DemoProgram");
    V24WriteLn("-1- Line 1");
    V24WriteLn("-2- Line 2");
    V24Write("make your choise :");
}

void main(void)
{
    char c;
    V24Init(9600);                   // init sci with 9600Baud, 8N1
    ShowUse();                       // display start msg
    while(1)                // loop ...
    {
        if (V24GetChar(&c)==TRUE)
        {
                if (c=='1')
                {
                        V24WriteLn("\n\n\rGreat! This was '1'");
                }
                else if (c=='2')
                {
                        V24WriteLn("\n\n\rSuper! '2'");
                }
                else
                {
                        V24WriteLn("\n\n\rSorry! Only '1' or '2' are supported!");
                }
                ShowUse();
        }
    }

}
```

## 4.3 PWM+SCI

On TOW (P76) you will see a PWM signal controlled by a host via V24.

```
/*-------------------------------------------------------------------
** PWM Output on TOW (P76) controlled by a host via V24
**-------------------------------------------------------------------*/

#include "mydefs.h"                  // with further includes!

unsigned char pwm_val = 75;          // set init dutycycle to 75%

/* some defines */
#define TIE         0x80
#define RIE         0x40
#define TE          0x20
#define RE          0x10
#define MPIE        0x08
#define TEIE        0x04

#define CK_INT      0x00
#define CK_INT_OUT  0x01
#define CK_EXT      0x02

#define IS_SCI_RDF      (SSR & 0x40)
#define CLEAR_SCI_RDF   SSR = (SSR & ~0x40)

#define IS_SCI_TX_FREE  (SSR & 0x80)

#define V24_BRR(x)    ((unsigned char)(((CPU_CLK+16*x)/32/x) - 1))


void V24Init (u16 Baudrate)
{
 SCR3 = 0x00;                        // disable all
 SSR = 0x00;                         // clear all errorbits
 SMR = 0x00;                         // 8N1 + /1 clock
 BRR = V24_BRR(Baudrate);            // set baud
 PMR1 |= 0x02;                       // P22 = TxD Output
 SCR3 = (TE|RE|CK_INT);              // Ints und Data disabled, internal clock
}

u8 V24NewChar(void)                  // check for new char on V24
{
 if (IS_SCI_RDF)                     // Receive buffer full?
 {
  return TRUE;
 }
 return FALSE;
}

u8 V24GetChar(u8* data)              // simple GetChar via V24
{
 if (IS_SCI_RDF)                     // Receive buffer full?
 {
  *data = RDR;                       // yes, get data
  CLEAR_SCI_RDF;                     // clear RDRF-Bit
  return TRUE;
 }
 return FALSE;
}


/*************************************
    TimerV-Test
    setup : CLK/8 (=1µs @ 8MHz)
                PWM-Output at TMOV (P76)
*************************************/
```

```
void Test_Timer_V(void)
{
    TCRV0 = 0x08|0x01;              // Clear by CompMatchA; IntClk/8
    TCRV1 = 0x01;                   // Clk/2, no external Trigger
    TCSRV = 0x08|0x01;              // 0=onCompMatchA, 1=onCompMatchB (output on P76)

    TCORA = 100;                    // set periode to 100 => 10.000Hz
    TCORB = pwm_val;                // set init dutycycle to 75%
}


void main(void)
{
    u8 c;
    V24Init(9600);                  // init sci with 9600Baud, 8N1
    Test_Timer_V();                 // for PWM-Output
    while(1)                        // do forever...
    {
        if (V24NewChar())
        {
                V24GetChar(&c);
                if (c == '+')
                {
                        if (pwm_val <100) pwm_val++;
                }
                else if (c == '-')
                {
                        if (pwm_val > 0) pwm_val--;
                }
                TCORB = pwm_val;            // set PWM-output
        }
    }
}
```

# Appendix A: CD-R content

## *Programs*

### IAR-EWH8 (Embedded workbench)

\programs\iar\

### HEW with IAR tool chain (Embedded workbench)

\programs\hew\

### FDT (flash development toolkit)

\programs\fdt\

## *Examples*

\examples\demoapp\

## *Datasheets*

Tiny Hitachi H8/3664F hardware manual, H8 programming manual, Tiny Application notes
\datasheets\

## *Documentation*

This manual as pdf
\documentation\

TiMo / TiBBo board schematic and layout
\documentation\schematic\

NOTES